

Oracle9i

Database Migration

Release 2 (9.2)

March 2002

Part No. A96530-01

ORACLE®

Oracle9i Database Migration, Release 2 (9.2)

Part No. A96530-01

Copyright © 1996, 2002 Oracle Corporation. All rights reserved.

Primary Author: Tony Morales

Graphic Artist: Valarie Moore

Contributors: Nipun Agarwal, Sanjay Agarwal, Rick Anderson, Vikas Arora, Rae Burns, Ben Chang, Lakshminaray Chidambaran, Eugene Chong, George Claborn, David Colello, Jay Davison, Alan Downing, Sreenivas Gollapudi, Brajesh Goyal, Tom Graves, Michael Hartstein, Jeffrey Hebert, Thuvan Hoang, Wei Huang, Robert Jenkins, Sanjeev Jhala, Christopher Jones, Mark Jungerman, Sanjay Kaluskar, Garrett Kaminaga, Dhiraj Kapoor, Vishwanath Karra, Mark Kennedy, Susan Kotsovolos, Viswanathan Krishnamurthy, Muralidhar Krishnaprasad, Paul Lane, Gordon Larimer, Simon Law, Jing Liu, Juan Loaiza, J. Bill Lee, Bill Maimone, Raghu Mani, Shailendra Mishra, Ari Mozes, Kannan Muthukkaruppan, Subramanian Muralidhar, Ravi Murthy, Karuna Muthiah, Mark Niebur, Peter Ogilvie, Naresh Pannani, Jenn Polk, Greg Pongracz, Franco Putzolu, N. C. Ramesh, Paul Raveling, Ann Rhee, Ajay Sethi, Carol Sexton, Helen Slattery, 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, except to the extent required to obtain interoperability with other independently created software or as specified by law, 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 ConText, Oracle Names, Oracle Store, Oracle7, Oracle8, Oracle8i, Oracle9i, PL/SQL, Pro*Ada, Pro*COBOL, Pro*C, Pro*C/C++, SQL*Net, and SQL*Plus are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Send Us Your Comments	xi
Preface.....	xiii
Audience	xiv
Organization.....	xiv
Related Documentation	xvi
Conventions.....	xvii
Documentation Accessibility	xxi
1 Introduction	
Terminology.....	1-2
Oracle Releases.....	1-2
Database Migration	1-3
Overview of Database Migration	1-3
Overview of Upgrade Steps.....	1-4
Role of the Database Administrator During the Upgrade	1-7
Role of the Application Developer During the Upgrade.....	1-8
Running Multiple Oracle Releases	1-9
Install Databases in Multiple Oracle Homes on the Same Computer	1-9
Install Databases in Multiple Oracle Homes on Separate Computers.....	1-10
Upgrade a Database to the Current Release.....	1-10
Upgrade Clients to the Current Release.....	1-10
Using Optimal Flexible Architecture (OFA).....	1-10
Changing Word Size.....	1-11

Rolling Upgrades	1-12
Deinstalling Options.....	1-12
2 Preparing to Upgrade	
Prepare to Upgrade	2-2
Become Familiar with the Features of the New Release	2-2
Determine Your Upgrade Path to the New Release	2-2
Choose an Upgrade Method	2-3
Choose an Oracle Home Directory for the New Release.....	2-8
Prepare a Backup Strategy.....	2-8
Develop a Testing Plan	2-8
Test the Upgrade Process	2-12
Test the Upgraded Test Database	2-12
3 Upgrading a Database to the New Oracle9i Release	
Install the Release 9.2 Oracle Software.....	3-2
Running the Database Upgrade Assistant Independently	3-4
Upgrade the Database Using the Database Upgrade Assistant.....	3-4
Upgrade the Database Manually	3-9
System Considerations and Requirements	3-9
Prepare the Database to be Upgraded.....	3-12
Upgrade the Database.....	3-15
Upgrading Specific Components	3-25
4 After Upgrading a Database	
Tasks to Complete After Upgrading Your Database	4-2
Back Up the Database	4-2
Change Passwords for Oracle-Supplied Accounts	4-2
Migrate Your Oracle Managed Files.....	4-2
Upgrade Oracle OLAP.....	4-4
Migrate Tables from LONGs to LOBs	4-6
Modify Your listener.ora File	4-8
Upgrade Your Standby Database.....	4-8
Add New Features as Appropriate.....	4-10

Develop New Administrative Procedures as Needed	4-10
Adjust Your Parameter File for the New Release	4-10
Normalize Filenames on Windows Operating Systems	4-11
Tasks to Complete Only After Upgrading a Release 8.1.7 or Lower Database	4-13
Tasks to Complete Only After Upgrading a Release 8.0.6 or Lower Database	4-16
Upgrading the Recovery Catalog.....	4-18
Upgrading Statistics Tables Created by the DBMS_STATS Package	4-19
Tasks to Complete Only After Upgrading a Release 7.3.4 Database	4-20
Test the Database and Compare Results	4-22
Tune the Upgraded Database	4-22
Changing the Word Size of Your Current Release	4-22

5 Compatibility and Interoperability

What Is Compatibility?.....	5-2
The COMPATIBLE Initialization Parameter	5-2
Setting the COMPATIBLE Initialization Parameter	5-7
Features Requiring a COMPATIBLE Setting	5-10
What Is Interoperability?	5-13
Compatibility and Interoperability Issues Between Release 9.2 and Release 9.0.1.....	5-14
Locally Managed SYSTEM Tablespace	5-14
Dictionary Managed Tablespaces	5-15
Change in Compatibility for Automatic Segment-Space Managed Tablespaces.....	5-15
Compatibility and Object Types.....	5-16
Oracle Managed Files.....	5-16
Oracle OLAP	5-16
Log Format Change with Parallel Redo	5-17
Oracle Dynamic Services	5-17
Oracle Syndication Server	5-17
Compatibility and Interoperability Issues Between Release 9.2 and Previous Releases...	5-18
Applications	5-19
The STARTUP Command	5-27
Tablespaces and Datafiles	5-27
Data Dictionary	5-29
Schema Objects	5-30
Datatypes.....	5-31

User-Defined Datatypes.....	5-34
SQL and PL/SQL.....	5-36
Advanced Queuing (AQ).....	5-37
Procedures and Packages	5-38
Oracle Optimizer.....	5-38
Oracle9i Real Application Clusters	5-39
Database Security.....	5-41
Database Backup and Recovery.....	5-43
Distributed Databases	5-47
SQL*Net or Oracle Net.....	5-49
Miscellaneous Compatibility and Interoperability Issues	5-51

6 Upgrading Your Applications

Overview of Upgrading Applications	6-2
Compatibility Issues for Applications	6-2
Upgrading Precompiler and OCI Applications	6-3
Understanding Software Upgrades and Your Client/Server Configuration	6-3
Compatibility Rules for Applications When Upgrading Oracle Software.....	6-4
Upgrading Options for Your Precompiler and OCI Applications	6-6
Upgrading SQL*Plus Scripts	6-9
Upgrading Oracle7 Forms or Oracle Developer Applications	6-10

7 Downgrading a Database Back to the Previous Oracle Release

Perform a Full Offline Backup	7-2
Remove Incompatibilities	7-2
Checking the Compatibility Level of Your Database	7-2
Identifying Incompatibilities.....	7-2
Removing Release 9.2 Incompatibilities	7-4
Removing Release 9.0.1 Incompatibilities	7-8
Reset Database Compatibility	7-20
Downgrade the Database	7-21

8 Database Migration Using Export/Import

Export Dump File Compatibility	8-2
---	------------

Export/Import Usage on Data Incompatible with a Previous Release	8-3
Source Database and Target Database	8-3
Export Utility Requirements	8-3
Import Utility Requirements	8-3
Upgrade the Source Database Using Export/Import	8-3

A Changes to Initialization Parameters and the Data Dictionary

Initialization Parameter Changes	A-2
Deprecated Initialization Parameters	A-2
Obsolete Initialization Parameters	A-3
Compatibility Issues with Initialization Parameters	A-5
New Default Value for DB_BLOCK_CHECKSUM	A-5
Maximum Number of Job Queue Processes	A-5
The ORACLE_TRACE_ENABLE Parameter	A-6
The SERIALIZABLE Parameter	A-6
SORT_AREA_SIZE and SORT_DIRECT_WRITES Parameters	A-6
New Default Value for LOG_CHECKPOINT_TIMEOUT	A-7
The O7_DICTIONARY_ACCESSIBILITY Parameter	A-7
The DML_LOCKS Parameter	A-7
The DB_DOMAIN Parameter	A-8
Parallel Execution Allocated from Large Pool	A-8
Archive Log Destination Parameters	A-11
Static Data Dictionary View Changes	A-14
Deprecated Static Data Dictionary Views	A-14
Obsolete Static Data Dictionary Views	A-16
Static Data Dictionary Views with Renamed Columns	A-16
Static Data Dictionary Views with Dropped Columns	A-17
Static Data Dictionary Views with Columns That May Return Nulls	A-18
Dynamic Performance View Changes	A-20
Deprecated Dynamic Performance Views	A-20
Obsolete Dynamic Performance Views	A-22
Dynamic Performance Views with Renamed Columns	A-23
Dynamic Performance Views with Dropped Columns	A-24

B Upgrade Considerations for Oracle Net Services

Overview of Unsupported Oracle Net Services Features	B-2
Unsupported Parameters and Control Utility Commands	B-4
Client and Database Coexistence Issues	B-4
Oracle9i Database Connections	B-4
Oracle8 or Oracle7 Database Connections	B-5
Oracle Names	B-6
Using the Oracle Net Manager to Handle Compatibility Issues	B-7
Upgrading to Oracle Net Services	B-8
Step 1: Verify Service Name and Instance Name.....	B-8
Step 2: Perform Software Upgrade on the Database Server	B-9
Step 3: Perform Software Upgrade on the Client.....	B-9
Step 4: Perform Functional Upgrade	B-9
Using Oracle Names Version 9	B-12
Upgrading from Oracle Names Version 2 Using a Database.....	B-13
Upgrading from Oracle Names Version 2 with the Dynamic Discovery Option	B-15
Upgrading from ROSFILES.....	B-17
Upgrading Region Checkpoint Files to Domain and Topology Checkpoint Files	B-19
Reviewing Upgrade Checklist	B-21

C Migrating from Server Manager to SQL*Plus

Startup Differences	C-2
Starting Server Manager	C-2
Starting SQL*Plus	C-2
Commands	C-3
Commands Introduced in SQL*Plus Release 8.1	C-3
Commands Common to Server Manager and SQL*Plus.....	C-4
SQL*Plus Equivalents for Server Manager Commands.....	C-5
Possible Differences in the SET TIMING Command.....	C-6
Server Manager Commands Unavailable in SQL*Plus.....	C-7
Syntax Differences	C-7
Comments	C-7
Blank Lines.....	C-9
The Hyphen Continuation Character	C-10
Ampersands.....	C-12

CREATE TYPE and CREATE LIBRARY Commands.....	C-13
COMMIT Command.....	C-14

D Upgrading an Oracle7 Database Using the MIG Utility

Overview of the MIG Utility	D-2
Outline of the Upgrade Process Using the MIG Utility.....	D-2
System Considerations and Requirements for Using the MIG Utility	D-3
Space Requirements	D-3
Block Size Considerations	D-4
Considerations for SQL*Net	D-4
Considerations for Replication Environments.....	D-5
Considerations for Migrating from ConText to Oracle Text.....	D-5
Distributed Database Considerations.....	D-5
Prepare the Oracle7 Database to be Upgraded	D-5
Review MIG Utility Command-Line Options	D-9
Run the MIG Utility	D-10
Run the MIG Utility on UNIX Operating Systems.....	D-10
Run the MIG Utility on Windows Platforms.....	D-12
Check the MIG Utility Results	D-13
Preserve the Oracle7 Database	D-14
MIG Utility Messages	D-14
Troubleshooting MIG Utility Errors	D-24
Problems Using the MIG Utility.....	D-25
Problems at the ALTER DATABASE CONVERT Statement	D-27
Abandoning the Oracle7 Upgrade	D-31
Migration Issues for Physical Rowids	D-32
Upgrading Applications and Migrating Data.....	D-33
The DBMS_ROWID Package	D-34
Snapshot Refresh	D-37
Oracle7 Client Compatibility Issues	D-37
ROWID Migration and Compatibility Issues.....	D-38
Changes to Initialization Parameters and the Data Dictionary in Release 8.0	D-39
Initialization Parameter Changes in Release 8.0	D-39
Static Data Dictionary View Changes in Release 8.0.....	D-40

E Database Migration and Compatibility for Replication Environments

Database Migration Overview for Replication	E-2
Upgrading All Sites at Once	E-3
Upgrading Incrementally	E-6
Preparing Oracle7 Master Sites for an Incremental Upgrade	E-7
Incremental Upgrade of Materialized View Sites.....	E-8
Incremental Upgrade of Master Sites	E-10
Upgrading to Primary Key Materialized Views	E-15
Primary Key Materialized View Conversion at Master Sites.....	E-16
Primary Key Materialized View Conversion at Materialized View Sites	E-16
Features Requiring an Upgrade to a Higher Release of Oracle	E-18
Features Requiring Oracle9i.....	E-18
Features Requiring Oracle8i or Higher.....	E-18
Features Requiring Oracle8 or Higher.....	E-19
Features That Work with Oracle7 and Higher Releases	E-19
Obsolete Procedures	E-20

Index

Send Us Your Comments

Oracle9i Database Migration, Release 2 (9.2)

Part No. A96530-01

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 database migrations on the Oracle database server. 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.

Oracle9i Database Migration contains information that describes the features and functionality of the Oracle9i (also known as the standard edition) and the Oracle9i Enterprise Edition products. Oracle9i and the Oracle9i 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 Oracle9i Real Application Clusters option.

See Also: *Oracle9i Database New Features* for information about the differences between Oracle9i and the Oracle9i 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

Oracle9i Database Migration is intended for database administrators (DBAs), application developers, security administrators, system operators, and anyone who plans or executes Oracle database migrations.

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

- Relational database concepts
- Your current release of the Oracle database server
- Your operating system environment

Organization

This document contains:

Chapter 1, "Introduction"

This chapter includes an overview of database migration as well as information about running multiple releases of Oracle. This chapter also provides information on changing the word size of your database during an upgrade or downgrade.

Chapter 2, "Preparing to Upgrade"

This chapter describes the steps to complete before upgrading a production database.

Chapter 3, "Upgrading a Database to the New Oracle9i Release"

This chapter guides you through the process of upgrading a database to the new Oracle9i release.

Chapter 4, "After Upgrading a Database"

This chapter describes the actions to complete after upgrading a database to the new Oracle9i release. This chapter also describes how to change the word size of your database (switching between 32-bit and 64-bit software).

Chapter 5, "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 features of Oracle along with their

required compatibility level and discusses specific issues relating to compatibility and interoperability.

Chapter 6, "Upgrading Your Applications"

This chapter provides general information about upgrading your applications and tools for use with the new Oracle9i release.

Chapter 7, "Downgrading a Database Back to the Previous Oracle Release"

This chapter guides you through the process of downgrading a database back to the previous Oracle release.

Chapter 8, "Database Migration Using Export/Import"

This chapter guides you through the process of using the Export and Import utilities to migrate data between Oracle databases.

Appendix A, "Changes to Initialization Parameters and the Data Dictionary"

This appendix lists changes to initialization parameters and the data dictionary across different releases of Oracle. This appendix also discusses compatibility issues with certain initialization parameters.

Appendix B, "Upgrade Considerations for Oracle Net Services"

This appendix describes coexistence and upgrade issues for Oracle Net Services.

Appendix C, "Migrating from Server Manager to SQL*Plus"

This appendix guides you through the process of modifying your Server Manager line mode scripts for use with SQL*Plus.

Appendix D, "Upgrading an Oracle7 Database Using the MIG Utility"

This appendix describes how to use the MIG utility to manually upgrade an Oracle7 database to the new Oracle9i release.

Appendix E, "Database Migration and Compatibility for Replication Environments"

This appendix provides step-by-step instructions for upgrading an Oracle Replication system on an Oracle7 database to Oracle9i. This appendix also discusses compatibility issues between different releases 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
- *Oracle9i 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://otn.oracle.com/admin/account/membership.html>

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://otn.oracle.com/docs/index.htm>

To access the database documentation search engine directly, please visit

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 for Microsoft Windows Operating Systems](#)

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 .
<i>Italics</i>	Italic typeface indicates book titles or emphasis.	<i>Oracle9i Database Concepts</i> Ensure that the recovery catalog and target database do <i>not</i> 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 <code>sqlplus</code> to open SQL*Plus. The password is specified in the <code>orapwd</code> file. Back up the datafiles and control files in the <code>/disk1/oracle/dbs</code> directory. The <code>department_id</code> , <code>department_name</code> , and <code>location_id</code> columns are in the <code>hr.departments</code> table. Set the <code>QUERY_REWRITE_ENABLED</code> initialization parameter to <code>true</code> . Connect as <code>oe</code> user. The <code>JRepUtil</code> class implements these methods.
lowercase monospace (fixed-width font) <i>italic</i>	Lowercase monospace italic font represents placeholders or variables.	You can specify the <i>parallel_clause</i> . Run <code>Uold_release.SQL</code> where <i>old_release</i> 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.	<code>DECIMAL (digits [, precision])</code>
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	<code>{ENABLE DISABLE}</code>
	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.	<code>{ENABLE DISABLE}</code> <code>[COMPRESS NOCOMPRESS]</code>

Convention	Meaning	Example
...	Horizontal ellipsis points indicate either: <ul style="list-style-type: none"> 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 	<pre>CREATE TABLE ... AS subquery; SELECT coll, col2, ... , coln FROM employees;</pre>
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	<pre>SQL> SELECT NAME FROM V\$DATAFILE; NAME ----- /fsl/dbs/tbs_01.dbf /fsl/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.</pre>
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	<pre>acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;</pre>
<i>Italics</i>	Italicized text indicates placeholders or variables for which you must supply particular values.	<pre>CONNECT SYSTEM/system_password DB_NAME = database_name</pre>
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.	<pre>SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;</pre>
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.	<pre>SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;</pre>

Conventions for Microsoft Windows Operating Systems

The following table describes conventions for Microsoft Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Choose Start >	How to start a program.	To start the Oracle Database Configuration Assistant, choose Start > Programs > Oracle - <i>HOME_NAME</i> > Configuration and Migration Tools > Database Configuration Assistant.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the file name begins with \\, then Windows assumes it uses the Universal Naming Convention.	c:\winnt "\system32 is the same as C:\WINNT\SYSTEM32
C:\>	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the <i>command prompt</i> in this manual. The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	C:\oracle\oradata> C:\>exp scott/tiger TABLES=emp QUERY=\"WHERE job='SALESMAN' and sal<1600\" C:\>impSYSTEM/passwordFROMUSER=scott TABLES=(emp, dept)
<i>HOME_NAME</i>	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	C:\> net start Oracle <i>HOME_NAME</i> TNSListener

Convention	Meaning	Example
<i>ORACLE_HOME</i> and <i>ORACLE_BASE</i>	<p>In releases prior to Oracle8i release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level <i>ORACLE_HOME</i> directory that by default used one of the following names:</p> <ul style="list-style-type: none"> ■ C:\orant for Windows NT ■ C:\orawin98 for Windows 98 <p>This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level <i>ORACLE_HOME</i> directory. There is a top level directory called <i>ORACLE_BASE</i> that by default is C:\oracle. If you install Oracle9i release 1 (9.0.1) on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is C:\oracle\ora90. The Oracle home directory is located directly under <i>ORACLE_BASE</i>.</p> <p>All directory path examples in this guide follow OFA conventions.</p> <p>Refer to <i>Oracle9i Database Getting Started for Windows</i> for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.</p>	Go to the <i>ORACLE_BASE\ORACLE_HOME\rdbms\admin</i> directory.

Documentation Accessibility

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

<http://www.oracle.com/accessibility/>

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

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Introduction

This chapter includes an overview of database migration as well as information about running multiple releases of Oracle. This chapter also provides information on changing the word size of your database during an upgrade or downgrade.

This chapter covers the following topics:

- [Terminology](#)
- [Overview of Database Migration](#)
- [Running Multiple Oracle Releases](#)
- [Using Optimal Flexible Architecture \(OFA\)](#)
- [Changing Word Size](#)
- [Rolling Upgrades](#)
- [Deinstalling Options](#)

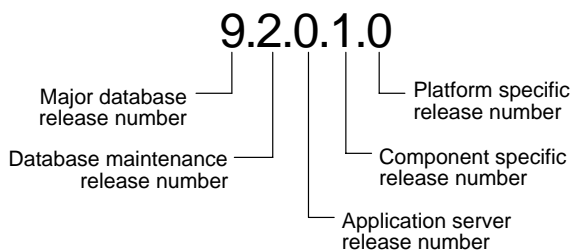
Terminology

The following terms are used throughout this document:

Oracle Releases

The instructions in this document describe moving between different **releases** of the Oracle database server. [Figure 1-1](#) describes what each part of a release number represents.

Figure 1-1 Example of an Oracle Release Number



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

Note: Starting with release 9.2, maintenance releases of Oracle are denoted by a change to the second digit of a release number. In previous releases, the third digit indicated a particular maintenance release.

When a statement is made in this book about a major database release number, the statement applies to all releases within that major database release. References to Oracle9i include all releases in release 9.0 and release 9.2; references to version 8 include all releases in release 8.0 and release 8.1. References to version 7 include all Oracle7 releases in release 7.0, release 7.1, release 7.2, and release 7.3.

Similarly, when a statement is made in this book about a maintenance release, the statement applies to all component specific and platform specific releases within that maintenance release. So, a statement about release 9.0.1 applies to release 9.0.1.1, release 9.0.1.1.2, and all other platform specific releases within release 9.0.1.

Database Migration

Database migration refers to the collection of processes and procedures for converting the data in an Oracle database to reflect a particular release of the Oracle database server. Database migration includes the following:

- The upgrade process, which upgrades a database to a new Oracle release.

See Also: [Chapter 3, "Upgrading a Database to the New Oracle9i Release"](#) for information about the upgrade process

- The downgrade process, which downgrades a database back to the Oracle release of the database prior to the upgrade.

See Also: [Chapter 7, "Downgrading a Database Back to the Previous Oracle Release"](#) for information about the downgrade process

Note: Since this book documents upgrading and downgrading between different releases of Oracle, this definition of database migration is appropriate. However, other Oracle documentation may use a broader definition of the term **migration**; for example, in some cases, migration may describe the process of moving data from a non-Oracle database into an Oracle database.

Overview of Database Migration

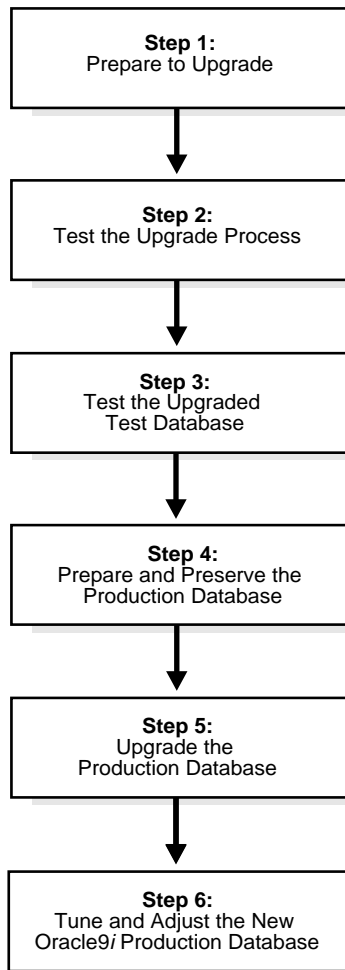
This section includes an overview of the major steps required to upgrade an existing Oracle database to the new Oracle9i release. Oracle9i is compatible with all earlier Oracle releases. Therefore, databases upgraded using the procedures described in this book can work in the same manner as in earlier releases and, optionally, can leverage new Oracle9i functionality.

Several preparatory steps are required before you upgrade your current production database. After the upgrade, you should perform several additional test steps. Other procedures enable you to add new Oracle9i functionality to existing applications.

Overview of Upgrade Steps

Before you upgrade a database, you should understand the major steps in the upgrade process. These 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.

Figure 1–2 Major Upgrade Steps



Careful planning and use of Oracle9i tools can ease the process of upgrading a database to the new Oracle9i release. You can use one of the following methods to upgrade a database:

- The Database Upgrade Assistant is the easiest way to upgrade a database. You can choose to run the Database Upgrade Assistant either during the installation of the new Oracle9i release or after installation is complete.

Note: The Database Upgrade Assistant is the preferred method of upgrading a database; Oracle Corporation highly recommends using the Database Upgrade Assistant to upgrade to the new Oracle9i release.

- A manual upgrade is more complicated but provides finer control over the process of upgrading a database.
- Export/Import and data copying enable piecemeal migration of parts of a database.

The following sections contain a brief outline of the major steps shown in [Figure 1-2](#). The purpose of these descriptions is to familiarize you with the major steps in the upgrade process. For detailed instructions, refer to the appropriate chapters in this book.

Step 1: Prepare to Upgrade

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

Step 2: Test the Upgrade Process

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

Step 3: Test the Upgraded Test Database

- Perform the tests you planned in Step 1 on the test database and on the test database that was upgraded to the new Oracle9i release.
- Compare results, noting anomalies between test results on the test database and on the upgraded 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 upgrade is completely successful and works with any required applications.

[Chapter 2, "Preparing to Upgrade"](#) provides detailed information about Steps 1 through 3.

Step 4: Prepare and Preserve the Production Database

- Prepare the current production database as appropriate to ensure that its upgrade to the new Oracle9i release will be successful.
- Schedule the downtime required for backing up and upgrading the production database.
- Perform a full backup of the current production database.

Step 5: Upgrade the Production Database

- Upgrade the production database to the new Oracle9i release.
- After the upgrade, perform a full backup of the production database and perform other post-upgrade tasks.

[Chapter 3](#) describes Steps 4 and 5 when using the Database Upgrade Assistant and when performing a manual upgrade. [Chapter 8](#) describes Steps 4 and 5 when using the Export/Import utilities. [Chapter 4](#) describes the backup procedure after the upgrade and other post-upgrade tasks.

See Also: [Appendix E, "Database Migration and Compatibility for Replication Environments"](#) if you are upgrading a database that has Oracle Replication installed

Step 6: Tune and Adjust the New Production Database

- Tune the new Oracle9i production database. The new Oracle9i production database should perform as good as, or better than, the database prior to the upgrade. [Chapter 4](#) describes these tuning adjustments.
- Determine which features of the new Oracle9i release you want to use and update your applications accordingly.
- Develop new database administration procedures as needed.
- Do not upgrade production users to the new Oracle9i database until all applications have been tested and operate properly. [Chapter 6](#) describes considerations for updating applications.

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

Role of the Database Administrator During the Upgrade

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

The specific DBA duties typically include the following:

- Meeting with everyone involved in the upgrade process and clearly defining their roles
- Performing test upgrades
- Scheduling the test and production upgrades
- Performing backups of the production database
- Completing the upgrade of the production database
- Performing backups of the newly upgraded Oracle9i production database

Role of the Application Developer During the Upgrade

The application developer is responsible for ensuring that applications designed for the current database work correctly with the upgraded Oracle9i database. Application developers often test applications against the upgraded Oracle9i database and decide which new features of Oracle9i should be used.

Before upgrading the production database, the DBA or application developer should install an Oracle9i test database. Then, the application developer can test and modify the applications, if necessary, until they work with their original (or enhanced Oracle9i) functionality.

The following references provide information about identifying differences in the upgraded Oracle9i database that could affect particular applications. Application developers can use these differences to guide modifications to existing applications.

- [Chapter 5, "Compatibility and Interoperability"](#) describes compatibility and interoperability issues that may result because of differences in releases of Oracle.
- [Chapter 6, "Upgrading Your Applications"](#) describes the changes required to enable existing applications to access an Oracle9i database and provides guidance for upgrading applications to take advantage of Oracle9i functionality.
- [Appendix A, "Changes to Initialization Parameters and the Data Dictionary"](#) lists obsolete and deprecated initialization parameters and data dictionary views.
- [Appendix B, "Upgrade Considerations for Oracle Net Services"](#) provides instructions for upgrading SQL*Net and Net8 to Oracle Net Services.
- [Appendix E, "Database Migration and Compatibility for Replication Environments"](#) provides instructions for upgrading a database that has Oracle Replication installed.
- *Oracle9i Database New Features* describes the features available in the new Oracle9i release
- *Oracle9i Real Application Clusters Concepts* and *Oracle9i SQL Reference* contain descriptions of changes and new Oracle9i 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:

- Oracle Net 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 releases.

Running Multiple Oracle Releases

You can run different releases of Oracle on the same computer at the same time. However, each release can only access a database that is consistent with its release. For example, if you have Oracle8*i* and Oracle9*i* installed on the same computer, then the Oracle8*i* server can access Oracle8*i* databases but not Oracle9*i* databases, and the Oracle9*i* server can access Oracle9*i* databases but not Oracle8*i* databases. The following sections provide general information about running multiple releases of Oracle.

Caution: It is not possible to install release 9.2 products into an existing Oracle home. This functionality was only available for certain previous releases and has not been continued. An Oracle9*i* release must be installed in a new 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 1-10 for more information.

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

Install Databases in Multiple Oracle Homes on the Same Computer

You can install Oracle7, Oracle8, Oracle8*i*, and Oracle9*i* databases in multiple (separate) Oracle homes on the same computer and have Oracle7, Oracle8, Oracle8*i*, and Oracle9*i* clients connecting to any or all of the databases.

Install Databases in Multiple Oracle Homes on Separate Computers

You can install Oracle7, Oracle8, Oracle8*i*, and Oracle9*i* databases in multiple (separate) Oracle homes on separate computers and have Oracle7, Oracle8, Oracle8*i*, and Oracle9*i* clients connecting to any or all of the databases.

Upgrade a Database to the Current Release

You can upgrade an Oracle7, Oracle8, Oracle8*i*, or Oracle9*i* database to the current Oracle9*i* release and have Oracle7, Oracle8, Oracle8*i*, and Oracle9*i* clients connecting to the upgraded database. You cannot upgrade the database in the same Oracle home.

Upgrade Clients to the Current Release

You can upgrade any or all of your Oracle7, Oracle8, Oracle8*i*, or Oracle9*i* clients to the current Oracle9*i* release. You can also upgrade your Oracle7, Oracle8, Oracle8*i*, or Oracle9*i* database to the current Oracle9*i* release at a later date.

Using Optimal Flexible Architecture (OFA)

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, isolates other fragmentation, and minimizes 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 operating system-specific Oracle documentation for more information about OFA
- *Oracle9i Database Administrator's Guide* for information about relocating database files

Changing Word Size

You can change the word size of your database after an upgrade or downgrade. 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.

Changing word size during an upgrade or downgrade is not supported. For example, if you have the 32-bit installation of release 9.2 on 32-bit hardware and you want to switch to the 64-bit installation of release 9.2 on 64-bit hardware, then complete the following steps:

1. Install the 32-bit installation of release 9.2 on 64-bit hardware.
2. Upgrade the database.
3. Follow the instructions in ["Changing the Word Size of Your Current Release"](#) on page 4-22.

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

- You can 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 switch from 64-bit hardware to 32-bit hardware, then you must first install 32-bit Oracle software.

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 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

The term **rolling upgrade** refers to upgrading different databases or different instances of the same database in Oracle9i Real Application Clusters one at a time, without stopping the database. Oracle9i Real Application Clusters does not support rolling upgrades.

Deinstalling Options

If you want to deinstall old options when you upgrade to the new Oracle9i release, then use the Oracle Universal Installer to deinstall them. You can deinstall them before or after you upgrade, but you must use the release of the installer that corresponds with the items you want to remove.

For example, if you are running release 9.0.1 with Oracle Text installed, and you decide that you do not need this option when you upgrade to the new Oracle9i release, then you should deinstall Oracle Text in one of the following ways:

- Before you upgrade to the new Oracle9i release, use the release 9.0.1 Oracle Universal Installer to deinstall Oracle Text. Then, do not install Oracle Text when you install the new Oracle9i release.
- When you upgrade to the new Oracle9i release, install and upgrade Oracle Text. Then, use the Oracle Universal Installer in the new Oracle9i release to deinstall Oracle Text.

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

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

Preparing to Upgrade

This chapter covers the steps that must be completed before you upgrade a production database. This chapter covers in detail Steps 1 through 3 of the upgrade process, which were outlined in "[Overview of Database Migration](#)" on page 1-3.

This chapter covers the following topics:

- [Prepare to Upgrade](#)
- [Test the Upgrade Process](#)
- [Test the Upgraded Test Database](#)

See Also:

- [Appendix B, "Upgrade Considerations for Oracle Net Services"](#) for information about upgrade considerations for Oracle Net Services
- [Appendix E, "Database Migration and Compatibility for Replication Environments"](#) if you are upgrading a database system that has Oracle Replication installed

Note: Some aspects of upgrading are operating system-specific. See your operating system-specific Oracle documentation for additional information about preparing to upgrade.

Prepare to Upgrade

Complete the following tasks to prepare to upgrade:

- [Become Familiar with the Features of the New Release](#)
- [Determine Your Upgrade Path to the New Release](#)
- [Choose an Upgrade Method](#)
- [Choose an Oracle Home Directory for the New Release](#)
- [Prepare a Backup Strategy](#)
- [Develop a Testing Plan](#)

Become Familiar with the Features of the New Release

Before you plan the upgrade process, become familiar with the features of the new Oracle9i release. *Oracle9i Database New Features* is a good starting point for learning the differences between Oracle releases. Also, check specific books in the Oracle9i documentation set to find information about new features for a certain component; for example, see *Oracle9i Real Application Clusters Concepts* for changes in Oracle9i Real Application Clusters.

Note: Oracle9i training classes are an excellent way to learn how to take full advantage of the functionality available with Oracle9i. Connect to the following Web page for more information:

<http://education.oracle.com>

Determine Your Upgrade Path to the New Release

The path that you must take to upgrade to the new Oracle9i release depends on the release of your database. [Table 2-1](#) contains the required upgrade path for each release of Oracle. Use the upgrade path and the specified documentation to upgrade your database.

Table 2–1 Upgrade Paths

Current Release	Upgrade Path
7.3.3 and Lower	<p>Direct upgrade is <i>not</i> supported. Complete the following steps to upgrade to the new release:</p> <ol style="list-style-type: none"> 1. Upgrade to release 7.3.4 using the instructions in release 7.3 of <i>Oracle7 Server Migration</i> and in the release 7.3.4 README. 2. Upgrade the release 7.3.4 database to the new release using the instructions in Chapter 3, "Upgrading a Database to the New Oracle9i Release" and Appendix D, "Upgrading an Oracle7 Database Using the MIG Utility".
7.3.4	<p>Direct upgrade is supported. Upgrade to the new release using the instructions in Chapter 3, "Upgrading a Database to the New Oracle9i Release" and Appendix D, "Upgrading an Oracle7 Database Using the MIG Utility".</p>
8.0.3 8.0.4 8.0.5	<p>Direct upgrade is <i>not</i> supported. Complete the following steps to upgrade to the new release:</p> <ol style="list-style-type: none"> 1. Upgrade to release 8.0.6 using the instructions in the release 8.0.6 <code>README.MIG.doc</code> file. 2. Upgrade the release 8.0.6 database to the new release using the instructions in Chapter 3, "Upgrading a Database to the New Oracle9i Release".
8.0.6	<p>Direct upgrade is supported. Upgrade to the new release using the instructions in Chapter 3, "Upgrading a Database to the New Oracle9i Release".</p>
8.1.5 8.1.6	<p>Direct upgrade is <i>not</i> supported. Complete the following steps to upgrade to the new release:</p> <ol style="list-style-type: none"> 1. Upgrade to release 8.1.7 using the instructions in <i>Oracle8i Migration</i>. 2. Upgrade the release 8.1.7 database to the new release using the instructions in Chapter 3, "Upgrading a Database to the New Oracle9i Release".
8.1.7 9.0.1	<p>Direct upgrade is supported. Upgrade to the new release using the instructions in Chapter 3, "Upgrading a Database to the New Oracle9i Release".</p>

If the release number of your database is not supported, then you must first upgrade your database to a supported Oracle release before upgrading to the new Oracle9i release.

Choose an Upgrade Method

Choose one of the following methods to upgrade your database to the new Oracle9i release:

- Use the Database Upgrade Assistant (DBUA).

The Database Upgrade Assistant can be launched by the Oracle Universal Installer, depending upon the type of installation that you select, and provides a graphical user interface (GUI) that guides you through the upgrade of a database. During installation, you can choose to not use the Database Upgrade Assistant, instead choosing to launch it as a standalone tool at any time in the future to upgrade a database.

- Perform a manual upgrade

A manual upgrade provides a command line upgrade of a database, using SQL scripts and utilities.

- Perform a full or partial export from your database, followed by a full or partial import into a new Oracle9i database.

Export/Import can copy a subset of the data in a database. Export/Import leaves the database unchanged, and makes a copy of the data.

- Copy data from a database into a new Oracle9i database using the SQL*Plus COPY command or the AS clause of the CREATE TABLE SQL statement.

Data copying can copy a subset of the data in a database. Data copying leaves the database unchanged, and makes a copy of the data.

The following sections describe each of the upgrade methods in detail, and discuss advantages and disadvantages of each method.

Database Upgrade Assistant

The Database Upgrade Assistant is a graphical user interface (GUI) tool that provides a simplified upgrade of a database to the new Oracle9i release. Online Help is available to assist you in its use.

The Database Upgrade Assistant performs all of the following pre-upgrade steps:

- It analyzes the database to be upgraded, determining which components of the database need upgrading. These components are then automatically upgraded after the database upgrade is complete.
- It checks the available space in the SYSTEM tablespace.
- It optionally backs up all necessary database files.
- It makes adjustments to the parameter file that are necessary for a successful upgrade. After the upgrade is complete, any initialization parameters that were temporarily adjusted are reverted back to their original values.

During the upgrade process, the Database Upgrade Assistant runs all necessary SQL scripts and utilities, removes obsolete initialization parameters and adjusts deprecated initialization parameters, and creates detailed logs for all SQL scripts and utilities executed during the upgrade.

When the upgrade is complete, the Database Upgrade Assistant provides a results dialog, describing all the details of the upgrade.

Starting with release 9.2, the Database Upgrade Assistant supports the upgrading of cluster databases.

Advantages of Using the Database Upgrade Assistant The following are some advantages of using the Database Upgrade Assistant:

- It filters all expected errors during the upgrade process.
- It ensures that sufficient resources are available.

The Database Upgrade Assistant performs several steps to accomplish the upgrade. It filters out all expected errors generated by the upgrade scripts.

If an unexpected error occurs during the upgrade, then the Database Upgrade Assistant gives you the option of skipping the current step and moving on to the next step of the upgrade. After the upgrade is complete, you can fix the cause of any errors and restart the Database Upgrade Assistant. The Database Upgrade Assistant resumes the upgrade by completing any steps that were skipped.

For example, if an unexpected error occurs during the upgrade of Oracle Spatial, then you can skip the Oracle Spatial upgrade and move on to the next component's upgrade. After the Database Upgrade Assistant has finished upgrading all components, you can restart the Database Upgrade Assistant to upgrade Oracle Spatial.

Manual Upgrade

A manual upgrade consists of running SQL scripts and utilities from a command line to upgrade a database to the new Oracle9i release.

When manually upgrading a database, you must perform the following pre-upgrade steps:

- Ensure that sufficient space in the SYSTEM tablespace exists, and add free space if it does not.
- Adjust your parameter file for the upgrade, to disable initialization parameters that might cause upgrade problems. After the upgrade is complete, any

initialization parameters that were modified must be reverted back to their original values.

You must also remove obsolete initialization parameters from your parameter file, and account for other initialization parameter changes, such as initialization parameters that have been deprecated.

- Perform a backup of the database.
- Some components of the database are not automatically upgraded when the database is upgraded. You must manually upgrade these components after the database upgrade is complete.

Depending on the release of the database being upgraded, you may need to perform additional pre-upgrade steps.

While a manual upgrade gives you finer control over the upgrade process, it is susceptible to error if any of the upgrade or pre-upgrade steps are either not followed or are performed out of order. The Database Upgrade Assistant performs all necessary pre-upgrade and upgrade steps.

Export/Import

Unlike the Database Upgrade Assistant or a manual upgrade, the Export/Import utilities physically copy data from your current database to a new database. The current database's Export utility copies specified parts of the database into an export dump file. Then, the Import utility of the new Oracle9i release loads the exported data into a new Oracle9i database. However, the new Oracle9i database must already exist before the export dump file can be copied into it.

When importing data from an earlier release, the Oracle9i Import utility makes appropriate changes to data definitions as it reads earlier releases' export dump files.

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

Export/Import Effects on Upgraded Databases The Export/Import upgrade method does not change the current database, which enables the database to remain available throughout the upgrade process. However, if a consistent snapshot of the database is required (for data integrity or other purposes), then the database must run in restricted mode or must otherwise be protected from changes during the export procedure. Because the current database can remain available, you can, for example, keep an existing production database running while the new Oracle9i database is being built at the same time by Export/Import. During the upgrade, to maintain

complete database consistency, changes to the data in the database cannot be permitted without the same changes to the data in the new Oracle9i database.

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

Export/Import Benefits Upgrading using 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 copying 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 current database.

Time Requirements for Export/Import Upgrading an entire database by using Export/Import can take a long time, especially compared to using the Database Upgrade Assistant or performing a manual upgrade. Therefore, you may need to schedule the upgrade during non-peak hours or make provisions for propagating to the new Oracle9i database any changes that are made to the current database during the upgrade.

Data Copying

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

Copying data and Export/Import offer the same advantages for upgrading. 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 copy 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 new database. Copying data is thus a good method for copying only part of a database table. In contrast, using Export/Import, you can copy only entire tables.

Choose an Oracle Home Directory for the New Release

You must choose an Oracle home directory for the new Oracle9i release that is separate from the Oracle home directory of your current release. You cannot install the new Oracle9i software into the same Oracle home directory as your current release.

Using separate installation directories enables you to keep your existing software installed along with the new Oracle9i software. This method enables you to test the upgrade process on a test database before replacing your production environment entirely.

Prepare a Backup Strategy

The ultimate success of your upgrade 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 Oracle9i databases

Develop a Testing Plan

You need a series of carefully designed tests to validate all stages of the upgrade process. Executed rigorously and completed successfully, these tests ensure that the

process of upgrading the production database is well understood, predictable, and successful. Perform as much testing as possible before upgrading the production database. Do not underestimate the importance of a test program.

The testing plan must include the following types of tests.

Upgrade Testing

Upgrade testing entails planning and testing the upgrade path from your current database to the new Oracle9i database, whether you use the Database Upgrade Assistant, perform a manual upgrade, or use Export/Import or other data-copying methods.

Regardless of the upgrade method you choose, you must establish, test, and validate an upgrade plan.

Minimal Testing

Minimal testing entails moving all or part of an application from the current database to the new Oracle9i database and running the application without enabling any new 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 immediately reveal any application startup or invocation problems.

Functional Testing

Functional testing is a set of tests in which new and existing functionality of the system are tested after the upgrade. Functional testing includes all database, networking, and application components. The objective of functional testing is to verify that each component of the system functions as it did before upgrading 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 a new Oracle9i database instance should be tested to ensure that there are no problems with the new software.
- Graphical user interfaces should be tested with other components.
- Subtle changes in the new Oracle9i 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 or not the application is directly connected to a new Oracle9i instance.

- If the connection between two components involves SQL*Net, Net8, or Oracle Net Services, then those connections should also be tested and stress tested.

Performance Testing

Performance testing of the new Oracle9i database compares the performance of various SQL statements in the new Oracle9i database with the statements' performance in the current database. Before upgrading, you should understand the performance profile of the application under the current database. Specifically, you should understand the calls the application makes to the database kernel.

For example, if you are using Oracle9i Real Application Clusters, and you want to measure the performance gains realized from using cache fusion when you upgrade to the new Oracle9i release, then make sure you record your system's statistics before upgrading. 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-cache fusion 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 Oracle9i release and evaluate your system's performance as described in *Oracle9i Real Application Clusters Deployment and Performance*.

See Also: *Oracle9i Database Performance Tuning 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 upgraded 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 upgrade the database, you should test the data to ensure that all data is accessible and that the applications function properly. You should also 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 production databases. Gather performance statistics for both normal and peak usage.

Specific Pre-Upgrade and Post-Upgrade 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 current database with the new Oracle9i database.

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

After upgrading, you can compare the execution plans of the new Oracle9i database with the execution plans of the current database. If there is a difference, then execute the statement on the new Oracle9i database and compare the performance with the performance of the statement executed on the current database.

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

Test the Upgrade Process

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

- If you plan to use the Database Upgrade Assistant or perform a manual upgrade, then create a test version (typically a subset) of the current production database to test the upgrade.
- If you plan to use Export/Import, then export and import small test pieces of the current production database.

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

Caution: Do not upgrade the actual production database until after you successfully upgrade 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 new Oracle9i database. Then, you can test these applications on a sample database before upgrading your current production database. See "[Upgrading Precompiler and OCI Applications](#)" on page 6-3 for more information.

Test the Upgraded Test Database

Perform the planned tests on the current database and on the test database that you upgraded to the new Oracle9i release. Compare the results, noting anomalies. Repeat the test upgrade as many times as necessary.

Test the newly upgraded Oracle9i test database with existing applications to verify that they operate properly with a new Oracle9i database. You also might test enhanced functionality by adding features that use the available Oracle9i functionality. However, first make sure that the applications operate in the same manner as they did in the current database.

See Also: [Chapter 6, "Upgrading Your Applications"](#) for more information on using applications with Oracle9i

Upgrading a Database to the New Oracle9i Release

This chapter guides you through the process of upgrading a database to the new Oracle9i release. This chapter covers the following topics:

- [Install the Release 9.2 Oracle Software](#)
- [Upgrade the Database Using the Database Upgrade Assistant](#)
- [Upgrade the Database Manually](#)

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.

Install the Release 9.2 Oracle Software

Complete the following steps to install the release 9.2 software:

1. If your operating system is UNIX, then make sure you are logged in as a user with write permission to the Oracle home and Oracle base directories, as well as all of their subdirectories.
2. Follow the instructions in your Oracle operating system-specific documentation to prepare for installation and start the Oracle Universal Installer.

If you are upgrading a cluster database, then see *Oracle9i Real Application Clusters Setup and Configuration* for additional installation instructions.

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. On Windows operating systems, enter the name of a new Oracle home in the Destination Name 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 9.2 release in a new Oracle home that is separate from the old release.

- d. Click Next.

The Available Products screen appears.

5. At the Available Products screen, select the Oracle9i server. The Oracle9i server is either Oracle9i Enterprise Edition or Oracle9i, depending on your installation medium. Then, click Next.
6. At the Installation Types screen, complete the following steps:

- a. If you are installing software only and will be performing an upgrade later, then select Enterprise Edition or Standard Edition. You should then select Software Only from the Database Configuration menu.
- b. Choose Custom if you would like finer control over the installation.

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, then 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 Database Upgrade Assistant, and if you are upgrading from Oracle7, the MIG utility.
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 Text in the previous database, then you should install Oracle Text in the new Oracle9i database.
 - b. If you are installing Oracle9i 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 an Existing Database screen.
7. At the Upgrading an Existing Database screen, complete the following steps:
- a. To upgrade a database using the Database Upgrade Assistant, select the Upgrade an Existing Database check box and choose the database to be upgraded.
To upgrade a database manually, or to run the Database Upgrade Assistant independently after installation is complete, do not select the Upgrade an Existing Database check box.
 - b. Click Next.

8. If the Create Database screen appears, then select the No option, indicating that you do not want to create a database because you are upgrading an existing database. Then, click Next.
9. 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.

When installation is complete, one or more assistants may be started. If you chose to run the Database Upgrade Assistant during installation, then you are ready to proceed with the upgrade When the Database Upgrade Assistant is started. See ["Upgrade the Database Using the Database Upgrade Assistant"](#) on page 3-4.

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

Running the Database Upgrade Assistant Independently

If you installed the new Oracle9i release without specifying that you are upgrading an existing database, then you can run the Database Upgrade Assistant independently after installation is complete.

Complete the following steps to run the Database Upgrade Assistant independently:

1. Start the Database Upgrade Assistant.

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

```
dbua
```

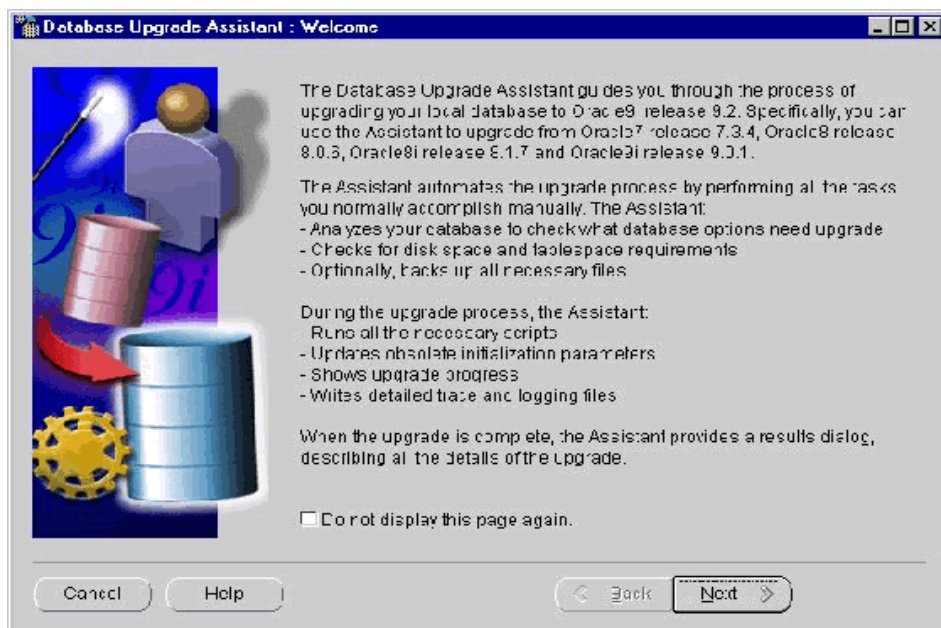
On Windows operating systems, choose:

```
Start > Programs > Oracle - HOME_NAME > Configuration and Migration Tools >  
Database Upgrade Assistant
```

You are ready to proceed with the upgrade when the Database Upgrade Assistant is started.

Upgrade the Database Using the Database Upgrade Assistant

When the Database Upgrade Assistant starts, its Welcome screen appears.

Figure 3–1 Welcome Screen of the Database Upgrade Assistant

Complete the following steps to upgrade a database using the Database Upgrade Assistant:

1. At the Welcome screen of the Database Upgrade Assistant (shown in [Figure 3–1](#)), make sure the database being upgraded meets the specified conditions. Then, click Next.

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

2. At the Select a Database screen, select the database you want to upgrade. Then, click Next.

You may need to provide a user name and password with SYSDBA privileges if you do not have operating system authentication.

3. If you are upgrading an Oracle7 database, then, at the Database Name screen, you can specify a new database name. Then, click Next.

4. At the Password screen, enter a valid password for each user listed. Then, click Next.

This screen only appears if the Database Upgrade Assistant requires a password for any user for the upgrade.

5. At the Backup screen, you have two options:
 - Choose "I have already backed up my database" if you completed a backup before running the Database Upgrade 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.

6. At the Network Configuration for the database screen, you have two tabs:

The Listeners tab is displayed if you have more than one listener in the release 9.2 Oracle home. Select the listeners in the release 9.2 Oracle home for which you would like to register the upgraded database.

The Directory Service tab shows up if you have directory service is configured in the release 9.2 Oracle home. You can select to either register or not register the upgraded database with the directory service.
7. At the Summary 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 Finish.

The Database Upgrade Assistant lists the initialization parameters that will be set for the database during the upgrade. The `COMPATIBLE` initialization parameter will be set to at least 8.1.0.

See Also: [Chapter 5, "Compatibility and Interoperability"](#) for information about setting the `COMPATIBLE` initialization parameter after the upgrade

8. A Progress dialog appears and the Database Upgrade Assistant begins to perform the upgrade.

You may encounter error messages with Ignore, Abort, and Skip the Step choices. If other errors appear, then you must address them accordingly. If an error is severe and cannot be handled during the upgrade, then you have the following choices:

- If Skip the step is presented as a choice in the message, then clicking the button will skip the current upgrade step.

This causes the Database Upgrade Assistant to move on to the next step in the upgrade, skipping this and any dependent steps. After the upgrade is complete, you can fix the problem, restart the Database Upgrade Assistant, and complete the skipped steps.

- If Skip the step is not presented as a choice in the message, then you need to abort the process by clicking the Abort button.

This will abort the upgrade process. The Database Upgrade Assistant prompts you to restore the database if the database backup was taken by the Database Upgrade Assistant.

After the database has been restored, you need to correct the cause of the error and restart the Database Upgrade Assistant to perform the upgrade again.

If you do not want to restore the database, then the Database Upgrade Assistant leaves the database in its present state so that you can proceed with a manual upgrade.

After the upgrade has completed, the following message is displayed on the Progress dialog:

```
Upgrade has been completed. Click the "OK" button to see the results of the upgrade.
```

Click the OK button.

9. At the Results dialog, you can view the details of the upgrade. You can also unlock and set passwords to the user accounts.

If you are not satisfied with the upgrade, then you can restore the database back to the previous release.

If you are satisfied with the upgrade, then click the Done button. The Database Upgrade Assistant removes the entry of the upgraded database from the old `listener.ora` file and reloads the listener of the old database.

- a. The Database Upgrade Assistant modifies the `SID_DESC` entry for the upgraded database in the Oracle9i `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/9.2`, then the assistant adds the following entry:

```
...
  (SID_DESC =
    (GLOBAL_DBNAME = sal.com)
    (ORACLE_HOME = /oracle/product/9.2)
    (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/9.2`, then the assistant adds the following entry:

```
...
  (SID_DESC =
    (GLOBAL_DBNAME = sal.com)
    (ORACLE_HOME = /oracle/product/9.2)
    (SID_NAME = SAL)
  )
...
```

This entry is the same as the entry in the simple case, but the Database Upgrade Assistant also adds the entry `an_entry` to the `SERVICE_NAMES` parameter in the `listener.ora` file. Therefore, the Database Upgrade Assistant changes the `SERVICE_NAMES` parameter to the following:

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

- b. The Database Upgrade Assistant removes the entry of the upgraded database from the old `listener.ora` file.
 - c. The Database Upgrade Assistant reloads the `listener.ora` file in both the old and new Oracle9i environments.
10. Complete the procedures described in [Chapter 4, "After Upgrading a 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 Oracle9i installation. Also, before you remove the old Oracle environment, make sure you relocate any datafiles in that environment to the new Oracle9i environment. See the *Oracle9i Database Administrator's Guide* for information about relocating datafiles.

Upgrade the Database Manually

Before you perform a manual upgrade, review the following system considerations and requirements.

System Considerations and Requirements

The following sections discuss system considerations and requirements.

Upgrading a Cluster Database

If you are upgrading a cluster database, then most of the actions described in this section should be performed on only one node of the system. So, perform the actions described in this section on only one node unless instructed otherwise in a particular step.

Migrating to a Different Operating System

You *cannot* migrate a database to a computer system that has a different operating system during the upgrade process. For example, you cannot migrate a database from Oracle7 on Solaris to Oracle9i on Windows 2000. However, you normally can use Export/Import to migrate a database to a different operating system.

Note: A change in word size is supported during the upgrade 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 1-11 for more information.

Considerations for Release 8.1.7 and Lower Database Character Sets

In Oracle9i, 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 (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 Database Globalization Support Guide* for more information about the Character Set Scanner

When you upgrade to Oracle9i, 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 Oracle9i 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 4-13.

Considerations for Replication Environments

If you plan to use `CHAR` column length semantics in Oracle9i, or if your replication database contains tables with `NCHAR` or `NVARCHAR2` columns, then this section contains considerations for upgrading a replication environment to Oracle9i.

CHAR Column Length Semantics If you plan to use `CHAR` column length semantics in a replication database after you upgrade it to Oracle9i, 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 Oracle9i databases if all of the databases prior to Oracle9i are using a single-byte character set. Otherwise, do not switch to `CHAR` column length semantics in the Oracle9i database until all of the other databases in the replication environment are upgraded to Oracle9i.

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 Oracle9i, 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 Oracle9i are using a fixed width national character set. If any of the databases prior to Oracle9i 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 Oracle9i.

See Also:

- *Oracle9i Replication* for more information about replication support for column length semantics and Unicode
- *Oracle9i Database 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

Prepare the Database to be Upgraded

Several preparatory steps are required before you upgrade your database to the new Oracle9i release. Depending on the release number of the database being upgraded, you may need to complete some or all of the following steps:

1. Review upgrade issues relating to SQL*Net, Net8, and Oracle Net Services.

See Also: [Appendix B, "Upgrade Considerations for Oracle Net Services"](#) for information

2. Log in to the system as the owner of the Oracle home directory of the database being upgraded.
3. Start SQL*Plus.

Note: If the database being upgraded is release 8.0.6 or lower, then start Server Manager. Do *not* start SQL*Plus.

4. Connect to the database instance as a user with SYSDBA privileges.
5. If the database being upgraded is release 8.0.6 or lower, then complete the following steps. Skip to Step 6 if the database being upgraded is release 8.1.7 or higher:
 - a. Make sure no user or role has the name OUTLN, because this schema is created automatically when you install Oracle9i. If you have a user or role named OUTLN, then you must drop the user or role and re-create it with a different name.

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.

- b. If the database being upgraded is release 7.3.4, then complete the additional preparatory steps in ["Prepare the Oracle7 Database to be Upgraded"](#) on page D-5.
6. 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.

[Table 3-1](#) identifies the amount of additional space in the SYSTEM tablespace required to upgrade to the new Oracle9i release from each supported Oracle release. If you run out of space during the upgrade, then you will need to perform the upgrade again.

Table 3-1 SYSTEM Tablespace Requirements

Release	Additional SYSTEM Tablespace	Additional SYSTEM Tablespace (with JServer)
9.0.1	16 MB	30 MB
8.1.7	52 MB	80 MB
8.0.6	70 MB	N/A
7.3.4	85 MB	N/A

The following example illustrates how to add more space to the SYSTEM tablespace:

```
ALTER TABLESPACE system
  ADD DATAFILE '/home/user1/mountpoint/oradata/db1/system02.dbf'
  SIZE 16M
  AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED;

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

7. Determine the files that you will back up in Step 10 by issuing the following SQL statements:

```
SQL> SPOOL files.log
SQL> SELECT member FROM v$logfile;
SQL> SELECT name FROM v$datafile;
SQL> SELECT name from v$controlfile;
SQL> SPOOL OFF
```

The `files.log` spool file lists all of the files that you must back up in Step 10.

8. Run `SHUTDOWN IMMEDIATE` on the database:

```
SQL> SHUTDOWN IMMEDIATE
```

If you are upgrading a cluster database, then shut down all instances.

9. 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.

10. Perform a full offline backup of the database. Make sure you back up the files listed in the `files.log` spool file that you generated in Step 7.

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

11. Exit SQL*Plus.

Upgrade the Database

Complete the following steps to upgrade the database:

1. If you are upgrading from release 7.3.4, then complete the following steps:
 - a. Complete the steps in "Review MIG Utility Command-Line Options" on page D-9 and in "Run the MIG Utility" on page D-10.
 - b. 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 upgrading a cluster database, then it may be set in the `initdb_name.ora` file instead.

You will issue the `ALTER DATABASE CONVERT` statement later in the upgrade process. This statement automatically creates new control files. If you do not use the `CONTROL_FILES` initialization parameter, then this statement uses the control file names of your previous 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 `dbfiles.log` spool file you created in Step 10 on page 3-14.

Control files are considerably larger in Oracle9i than in Oracle7. For example, Oracle7 control files in the hundreds of kilobytes may expand into tens of megabytes in Oracle9i. The larger size in Oracle9i 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 Oracle9i Real Application Clusters, perform this step on all nodes.

2. If your operating system is Windows, then complete the following steps:
 - a. Stop the OracleServiceSID Oracle service of the database you are upgrading, where SID is the instance name. For example, if your SID is ORCL, then enter the following at a command prompt:

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

- b. Delete the Oracle service at a command prompt using ORADIM. The following table lists the command to run for each Oracle release:

Oracle Release...	Enter at a Command Prompt...
7.3	C:\> ORADIM73 -DELETE -SID SID
8.0	C:\> ORADIM80 -DELETE -SID SID
8.1 and higher	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 command:

```
C:\> ORADIM80 -DELETE -SID ORCL
```

If your Oracle release is release 8.1.7 and your SID is ORCL, then enter the following command:

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

- c. Create the new Oracle9i database service at a 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.

<i>PASSWORD</i>	is the password for the new release 9.2 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.
<i>USERS</i>	is the maximum number of users who can be granted SYSDBA and SYSOPER privileges.
<i>ORACLE_HOME</i>	is the release 9.2 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:\ORA92, then enter the following command:

```
C:\> ORADIM -NEW -SID ORCL -INTPWD TWxy579 -MAXUSERS 10
      -STARTMODE AUTO -PFILE C:\ORA92\DATABASE\INITORCL.ORA
```

3. Copy configuration files from the previous Oracle home to the new Oracle9i Oracle home:
 - a. If you are upgrading from release 7.3.4 on a UNIX platform, then move or copy the convert file from the Oracle7 Oracle home directory to the new Oracle9i Oracle home directory. On most UNIX platforms, the convert file, *convsid.dbf* (where *sid* is the Oracle9i database name), should reside in *ORACLE_HOME/dbs* in both the Oracle7 and the new Oracle9i environment.

On Windows operating systems, the convert file, *convert.ora*, should reside in *ORACLE_HOME\rdbms* in the new Oracle9i environment. It is automatically placed in this directory by the MIG utility, and you do not need to move it.
 - b. If your parameter file resides within the old environment's Oracle home, then copy it to the new Oracle home. By default, Oracle looks for the parameter file in *ORACLE_HOME/dbs* on UNIX platforms and in *ORACLE_HOME\database* on Windows operating systems. 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 Oracle9i release.
 - c. If your 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 the new Oracle

home. The file specified in the `IFILE` entry contains additional initialization parameters.

- d. If you have a password file that resides within the old environment's Oracle home, then move or copy the password file to the new Oracle9i Oracle home.

The name and location of the password file are operating system-specific. On UNIX platforms, the default password file is `ORACLE_HOME/dbs/orapwsid`. On Windows operating systems, the default password file is `ORACLE_HOME\database\pwdsid.ora`. In both cases, `sid` is your Oracle instance ID.

- e. If you are upgrading a cluster database and your `initdb_name.ora` file resides within the old environment's Oracle home, then move or copy the `initdb_name.ora` file to the new Oracle home.

Note: If you are upgrading a cluster database, then perform this step on all nodes in which this cluster database has instances configured.

4. Adjust your parameter file for use with the new Oracle9i release by completing the following steps:
 - a. Remove obsolete initialization parameters and adjust deprecated initialization parameters. Certain initialization parameters are obsolete in the new Oracle9i release. Remove all obsolete parameters from any parameter file that will start a release 9.2 instance. Obsolete parameters may cause errors in the new Oracle9i release. Also, alter any parameter whose syntax has changed in the new Oracle9i release; refer to [Appendix A, "Changes to Initialization Parameters and the Data Dictionary"](#) for lists of initialization parameters that have been deprecated or have become obsolete.

Also, if you are upgrading a cluster database, then see *Oracle9i Real Application Clusters Setup and Configuration* for more information about obsolete cluster database initialization parameters.

- b. Make sure the `COMPATIBLE` initialization parameter is properly set for Oracle9i. If `COMPATIBLE` is set below 8.1.0, then you will encounter the following error when you attempt to start up your release 9.2 database later in step 10:

```
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`.

- c. 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.
- d. If you are upgrading a cluster database, then set the `CLUSTER_DATABASE` initialization parameter to `false`. After the upgrade, you must set this initialization parameter back to `true`.
- e. Make sure the `DB_DOMAIN` initialization parameter is set properly.

See Also: ["The DB_DOMAIN Parameter"](#) on page A-8 for more information about setting this initialization parameter.

- f. If the `NLS_LENGTH_SEMANTICS` initialization parameter is set to `CHAR`, then set it to `BYTE`. This initialization parameter can be set back to `CHAR` after the upgrade is complete.
- g. On Windows operating systems, 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
<code>BACKGROUND_DUMP_DEST</code>	<code>ORACLE_BASE\oradata\DB_NAME</code>
<code>USER_DUMP_DEST</code>	<code>ORACLE_BASE\oradata\DB_NAME\archive</code>

In the settings, substitute the complete Oracle base path for `ORACLE_BASE` and substitute the database name for `DB_NAME`.

- h. Make sure all path names in the parameter file are fully specified. You should not have relative path names in the parameter file.
- i. If the parameter file contains an `IFILE` entry, then change the `IFILE` entry in the parameter file to point to the new location of the include file that you specified in Step 3. c. Then, edit the file specified in the `IFILE` entry in the same way that you edited the parameter file in Steps a to h.

- j. If you are upgrading a cluster database, then modify the `initdb_name.ora` file in the same way that you modified the parameter file.

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

Note: If you are upgrading a cluster database, then perform this step on all nodes in which this cluster database has instances configured.

5. If your operating system is UNIX, then make sure that the following environment variables point to the new release 9.2 directories:
 - `ORACLE_HOME`
 - `PATH`
 - `ORA_NLS33`
 - `LD_LIBRARY_PATH`

If you are upgrading from release 7.3.4 and `ORACLE_HOME` points to the Oracle7 executables, then the following error is displayed when you issue the `ALTER DATABASE CONVERT` statement later in the upgrade process:

```
ORA-00223: convert file is invalid or incorrect version
```

Note: If you are upgrading a cluster database, then perform this step on all nodes in which this cluster database has instances configured.

See Also: Your operating system-specific Oracle9i installation documents for information about setting other important environment variables on your operating system.

6. Log in to the system as the owner of the Oracle home directory of the new release.
7. At a system prompt, change to the `ORACLE_HOME/rdbms/admin` directory.
8. Start SQL*Plus.
9. Connect to the database instance as a user with SYSDBA privileges.

10. If the database being upgraded is release 8.0.6 or higher, then Start up the database by issuing the following command:

```
SQL> STARTUP MIGRATE
```

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.

11. If the database being upgraded is release 7.3.4, then perform the following steps to Start up the database:
 - a. Start an Oracle9i database instance without mounting the new Oracle9i 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 upgrade normally. Then, remove the obsolete initialization parameters the next time you shut down the database.

- b. Create a new Oracle9i database control file and convert the file headers of all online tablespaces to Oracle9i 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 MIG utility. Otherwise restore the backup you performed after you ran the MIG utility.

See Also: ["Problems at the ALTER DATABASE CONVERT Statement"](#) on page D-27 for information about common errors encountered at this step and the actions required to resolve them.

- c. Open the Oracle9i database with the following statement:

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

When the Oracle9i database is opened, all rollback segments that are online are converted to the new Oracle9i 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, but make sure you completed all of the pre-migration steps described in [Chapter 2](#).

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

```
SQL> SPOOL upgrade.log
```

13. If you want to see the complete detailed output of the scripts you will run, then you can issue a `SET ECHO ON` command:

```
SQL> SET ECHO ON
```

14. Run `uold_release.sql`, where `old_release` refers to the release you had installed prior to upgrading. See [Table 3-2](#) 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 3-2 Upgrade Scripts

Old Release	Run Script
7.3.4	u0703040.sql
8.0.6	u0800060.sql
8.1.7	u0801070.sql
9.0.1	u0900010.sql

Note: If the old release you had installed prior to upgrading is not listed in [Table 3-2](#), 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 9.2 installation.
- You must run the script in the new release 9.2 environment.
- You only need to run one script, even if your upgrade spans more than one release. For example, if your old release was 8.1.7, then you only need to run `u0801070.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 new 9.2 release, which create the system catalog views and all the necessary packages for using PL/SQL.

15. 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 `upgrade.log`. Correct any problems you find in this file and rerun the appropriate upgrade script if necessary. You can rerun any of the scripts described in this chapter as many times as necessary.

16. To identify which components were loaded into the database for the previous release, display the contents of the registry after the upgrade script completes. All of the components displayed need to be upgraded to release 9.2:

```
SQL> SELECT comp_id, version, status
        FROM dba_registry;
```

17. Run the component upgrade script to upgrade components whose upgrades can be run while connected with SYSDBA privileges:

```
SQL> @cmpdbmig.sql
```

In a separate session, verify that the component upgrades ran successfully by reviewing the `cmp_upgrade.log` file, and then rerunning `cmpdbmig.sql` if necessary.

18. If you issued a `SET ECHO ON` command, then you may want to issue a `SET ECHO OFF` command now:

```
SQL> SET ECHO OFF
```

19. Shut down and restart the instance to reinitialize the system parameters for normal operation. The restart will also perform release 9.2 initialization for JServer and other components.

```
SQL> SHUTDOWN IMMEDIATE
```

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 Oracle9i database.

Also, if you encountered a message listing obsolete initialization parameters when you started the database in Step 10, then remove the obsolete initialization parameters from the initialization parameter file now.

20. Upgrade any remaining components that existed in the previous database.
21. Run `utlrlp.sql` to recompile any remaining stored PL/SQL and Java code.

```
SQL> @utlrlp.sql
```

Verify that all expected packages and classes are valid:

```
SQL> SELECT count(*) FROM dba_objects WHERE status='INVALID';  
SQL> SELECT UNIQUE name FROM dba_objects WHERE status='INVALID';
```

Verify that all components are valid and have been upgraded to release 9.2:

```
SQL> SELECT comp_id, version, status FROM dba_registry;
```

Your database is now upgraded to the new 9.2 release. Complete the procedures described in ["Upgrading Specific Components"](#) on page 3-25 and in [Chapter 4, "After Upgrading a Database"](#).

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 9.2 installation directory. Also, before you remove the old Oracle environment, make sure you relocate any datafiles in that environment to the new Oracle9i environment. See the *Oracle9i Database Administrator's Guide* for information about relocating datafiles.

Upgrading Specific Components

Some components of the Oracle database server require an upgrade separate from the integrated component upgrades performed by `cmpdbmig.sql`. [Table 3-3](#) lists components and their upgrade status:

Table 3-3 Component Upgrade Status

Installed Component	Automatically Upgraded
Oracle9i Catalog Views	Yes
Oracle9i Packages and Types	Yes
JServer JAVA Virtual Machine	Yes
Oracle9i Java Packages	Yes
Oracle XDK for Java	Yes
Messaging Gateway	Yes
Oracle Text	No
Oracle9i Real Application Clusters	Yes
Oracle Workspace Manager	Yes
Oracle Data Mining	Yes
Oracle Ultra Search	No
OLAP Catalog	Yes
Oracle Spatial	No
Oracle interMedia	No
Oracle Visual Information Retrieval	No
Oracle Label Security	Yes

Complete the actions in the following sections to upgrade components that were not automatically upgraded.

Note: You should perform the actions described in these sections only after you have upgraded the database by following the instructions earlier in this chapter.

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 9.2.

Upgrading Oracle *interMedia*

Upgrade instructions for Oracle *interMedia* 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.

Upgrading Oracle Visual Information Retrieval

Upgrade instructions for Oracle Visual Information Retrieval 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 text_upgrade.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` command:

```
SQL> SET ECHO ON
```

8. If you are upgrading from release 8.1.7, then complete the following steps. Skip to Step 9 if you are upgrading from release 9.0.1.

- 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
```

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

9. If you are upgrading from release 9.0.1, then complete the following steps.

- a. Run `s0902000.sql`:

```
SQL> @s0902000.sql
```

This script grants new, required database privileges to user `CTXSYS`.

- b. Connect to the database instance as user `CTXSYS`.

- c. Run `u0902000.sql`:

```
SQL> @u0902000.sql
```

This script upgrades the `CTXSYS` schema to release 9.2.

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

10. Check for any invalid `CTXSYS` objects and alter compile as needed.

11. 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 `text_upgrade.log`. Correct any problems you find in this file and rerun the appropriate upgrade scripts if necessary.

If you issued a `SET ECHO ON` command, then you may want to issue a `SET ECHO OFF` command now:

```
SQL> SET ECHO OFF
```

12. Run ALTER SYSTEM DISABLE RESTRICTED SESSION:

```
SQL> ALTER SYSTEM DISABLE RESTRICTED SESSION;
```

13. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

14. Exit SQL*Plus.

Oracle Text is upgraded to the new release.

Upgrading Oracle Ultra Search

If the Oracle system has Oracle Ultra Search installed, then see the *Oracle Ultra Search Online Documentation* for instructions about upgrading Oracle Ultra Search to release 9.2.

After Upgrading a Database

This chapter guides you through the procedures to perform after you have completed an upgrade of your database. This chapter covers the following topics:

- [Tasks to Complete After Upgrading Your Database](#)
- [Test the Database and Compare Results](#)
- [Changing the Word Size of Your Current Release](#)

Tasks to Complete After Upgrading Your Database

Complete the following tasks after you have 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 a database

Change Passwords for Oracle-Supplied Accounts

Depending on the release from which you 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;
```

Migrate Your Oracle Managed Files

If you are upgrading from an Oracle9i release earlier than release 9.0.1.2.0, then you must migrate your Oracle Managed Files. In Oracle9i releases earlier than release 9.0.1.2.0, Oracle sometimes incorrectly considered non-OMF files as OMF. This resulted in the following error when adding a datafile, control file, or log file to the database:

```
ORA-01276: Cannot add a file with an Oracle Managed Files file name
```

Also, Oracle sometimes incorrectly deleted the operating system files associated with a tablespace or redo log when dropping the tablespace or redo log.

Starting with release 9.0.1.2.0, the format of Oracle Managed Files file names on Windows and UNIX operating systems has changed. OMF files created in earlier Oracle9i releases will not be recognized as OMF files unless they are renamed to conform to the new OMF file name format.

In earlier Oracle9i releases, a file was considered OMF if its base file name contained:

- An `ora_` prefix
- A `.dbf`, `.tmp`, `.log`, or `.ctl` extension

In release 9.0.1.2.0 and higher, a file is now considered OMF if its base file name contains:

- An `ol_mf_` prefix
- A `.dbf`, `.tmp`, `.log`, or `.ctl` extension
- An underscore (`_`) immediately preceding the extension

You can migrate old OMF datafiles, tempfiles, and log files by renaming them in the file system and in the control file. Complete the following steps:

1. Find the OMF files by issuing the following SQL statements:

```
SQL> SELECT name FROM v$datafile;
SQL> SELECT name FROM v$tempfile;
SQL> SELECT member FROM v$logfile;
```

2. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

3. Rename the files in the file system:

- Change `ora_` to `ol_mf_`
- Add `_` before the extension

For example, for a file named `ora_tbs1_2ixfh90q.dbf`, the new name would be `ol_mf_tbs1_2ixfh90q_.dbf`.

4. Mount the database.
5. Rename the files in the control file. For example:

```
SQL> ALTER DATABASE RENAME FILE 'old_filename' TO 'new_omf_filename';
```

6. Open the database.

OMF control files can be migrated by renaming them in the file system and in the `CONTROL_FILES` initialization parameter. Complete the following steps:

1. Find the OMF files by examining the `CONTROL_FILES` initialization parameter.
2. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

3. Rename the files in the file system:

- Change `ora_` to `ol_mf_`
- Add `_` before the extension

For example, for a file named `ora_cmr7t90p.ctl`, the new name would be `ol_mf_cmr7t90p_.ctl`.

4. Modify the `CONTROL_FILES` initialization parameter to reference the new names.
5. Mount and open the database.

Upgrade Oracle OLAP

This section contains Oracle OLAP upgrade instructions.

Upgrading from Release 8.1.7 or Later

Oracle OLAP provides access to analytic workspaces through SQL. If your `COMPATIBLE` initialization parameter is set to 8.1.6 or higher, then the standard upgrade procedure provides this functionality. No additional steps are required.

Oracle OLAP also offers the OLAP API (a Java interface) and the OLAP Catalog Metadata. To include these features when `COMPATIBLE` is 8.1.6 or higher, perform the following steps:

1. Complete the standard upgrade procedure.
2. Set `COMPATIBLE` to 9.2.0.
3. Restart the database.
4. Run the following script:

```
ORACLE_HOME/olap/admin/olapapi.sql
```

5. If you are upgrading from a release that is earlier than release 9.0.1, then complete the following additional steps which create the OLAP Catalog Metadata in its own tablespace:
 - a. Create a tablespace with a statement similar to the following. You can specify any valid tablespace name and any valid database file name:


```
CREATE TABLESPACE OLAPCAT LOGGING
  DATAFILE 'ORACLE_HOME/rdbms/dbs/olap01.dbf'
  SIZE 20M REUSE AUTOEXTEND ON NEXT 640K
  MAXSIZE UNLIMITED EXTENT MANAGEMENT LOCAL
  SEGMENT SPACE MANAGEMENT AUTO;
```
 - b. Run the following script specifying the name of the tablespace that you created and the name of a temporary tablespace for your database. In this example, the names are OLAPCAT and TEMP:


```
ORACLE_HOME/cwmlite/admin/oneinst1.sql OLAPCAT TEMP
```

Upgrading from Release 8.0.6 or Earlier

If you only want access to analytic workspaces through SQL, without the OLAP API and the OLAP Catalog Metadata, then complete the following steps:

1. Complete the standard upgrade procedure.
2. Set `COMPATIBLE` to 8.1.6 or higher.
3. Restart the database.
4. Run the following script:

```
ORACLE_HOME/olap/admin/olapaw.sql
```

If you want support for the OLAP API and OLAP Catalog metadata in addition to analytic workspace access through SQL, then complete the following steps instead:

1. Complete the standard upgrade procedure.
2. Set `COMPATIBLE` to 9.2.0.
3. Restart the database.
4. Run the following script:

```
ORACLE_HOME/olap/admin/olap.sql
```

5. Create a tablespace with a statement similar to the following. You can specify any valid tablespace name and any valid database file name:

```
CREATE TABLESPACE OLAPCAT LOGGING
  DATAFILE 'ORACLE_HOME/rdbms/dbs/olap01.dbf'
  SIZE 20M REUSE AUTOEXTEND ON NEXT 640K
  MAXSIZE UNLIMITED EXTENT MANAGEMENT LOCAL
  SEGMENT SPACE MANAGEMENT AUTO;
```

6. Run the following script specifying the name of the tablespace that you created and the name of a temporary tablespace for your database. In this example, the names are OLAPCAT and TEMP:

```
ORACLE_HOME/cwmlite/admin/oneinst1.sql OLAPCAT TEMP
```

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 Oracle9i, 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.

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 Database Upgrade Assistant to upgrade your database.
- You used the Database Upgrade Assistant to 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 B-11 for information about modifying your `listener.ora` file.

Upgrade Your Standby Database

The following procedures contain information about upgrading your current release of Oracle to the new Oracle9i release for a configuration that includes one or more standby databases.

Prepare to Upgrade

If multiple standby databases exist, then repeat the steps in this section for each standby database to be upgraded:

1. Check for the existence of nologging operations. If nologging operations have been performed, then the standby will need to be updated. Refer to *Oracle9i Data Guard Concepts and Administration* for further details.
2. Make note of any tablespaces or datafiles that need recovery due to offline immediate. Tablespaces or datafiles should be recovered and either brought online or taken offline prior to upgrading.

Upgrade the Production Site

Install the new Oracle9i release on production sites and follow the instructions in Oracle9i for upgrading the production database.

Make the following additional adjustments to your parameter file before the upgrade:

- Do not enable remote archiving within the production database's parameter file if it was not already enabled. If remote archiving is enabled, then set the remote destination to defer.
- Cancel managed recovery on the standby database if running.
- If upgrading from release 8.1.7 or earlier and running Oracle9i Real Application Clusters Guard, make sure to comment out the `PARALLEL_SERVER` initialization parameter and set `CLUSTER_DATABASE = true` on the production site.

Ensure that all archived redo logs have been applied to the standby prior to the upgrade.

After the upgrade is complete, switch logfiles to archive any redo that remains in the last log:

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

Manually transfer archive logs from the upgrade from the primary archive destination on the production site to the standby archive destination on the standby host.

Shut down the standby database and listener

Start up and mount the standby database.

Place the standby database in managed recovery mode. At the SUGGESTION prompt, type AUTO to apply all of the archive logs generated during the upgrade process.

Verify that the standby database has been recovered to the last log that was transferred to the standby host. Resolve any archive log gaps between the production and the standby.

Re-enable remote archiving on the primary site by changing the standby destination from defer to enable.

Place standby into a recovery state.

Add New Features as Appropriate

Oracle9i Database New Features describes many of the new features available in the new Oracle9i release. 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 new Oracle9i database. You may prefer to introduce these enhancements into your database and corresponding applications gradually.

[Chapter 6, "Upgrading Your Applications"](#) describes ways to enhance your applications so that you can take advantage of new Oracle9i features. However, before you implement new Oracle9i features, test your applications and successfully run them with the upgraded database.

Develop New Administrative Procedures as Needed

After familiarizing yourself with new Oracle9i 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 Parameter File for the New Release

Each release of Oracle introduces new initialization parameters, deprecates some initialization parameters, and makes some initialization parameters obsolete. You should adjust your parameter file to account for these changes and to take advantage of new initialization parameters that may be beneficial to your system.

See Also:

- *Oracle9i Database Reference* for a list of the new initialization parameters in release 9.2, and for information about each parameter
- [Appendix A, "Changes to Initialization Parameters and the Data Dictionary"](#) for lists of obsolete and deprecated initialization parameters in release 9.2

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

See Also: ["Setting the COMPATIBLE Initialization Parameter"](#) on page 5-7 for information

Normalize Filenames on Windows Operating Systems

You only need to normalize filenames if you are running Oracle on a Windows operating system. You do not need to perform these steps on UNIX platforms.

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 operating systems, 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.RENORMALIZEALLFILENAME`s procedure to normalize the filenames in your control file:

```
SQL> EXECUTE DBMS_BACKUP_RESTORE.RENORMALIZEALLFILENAME;
```

6. When the `DBMS_BACKUP_RESTORE.RENORMALIZEALLFILENAME`s 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 through 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 Upgrading a Release 8.1.7 or Lower Database

Complete the following additional tasks only if you upgraded your database from release 8.1.7 or lower. These tasks are *not* required if you upgraded from release 9.0.1.

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 Oracle9i.

The following steps convert your `NCHAR` columns from the old format and character set to the new Oracle9i format. In addition, if your old National Character Set was UTF8, it will remain UTF8 in Oracle9i. 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 Oracle9i AL16UTF16 National Character Set or a version 8 non-UTF8 National Character Set can be converted to an Oracle9i UTF8 National Character Set.

You will encounter the following error when attempting to use the NCHAR columns in Oracle9i 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 Oracle9i, 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 Server Manager Line Mode Scripts to SQL*Plus

Oracle9i 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. [Appendix C, "Migrating from Server Manager to SQL*Plus"](#) contains instructions for modifying your Server Manager line mode scripts to work with SQL*Plus.

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 Oracle9i 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 Setup 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

Tasks to Complete Only After Upgrading a Release 8.0.6 or Lower Database

Complete the following additional tasks only if you upgraded your database from release 8.0.6 or lower. These tasks are *not* required if you upgraded from release 8.1.7 or higher.

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 A-8 for information about adjusting these parameters.

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.

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 5-43.

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.

Tasks to Complete Only After Upgrading a Release 7.3.4 Database

Complete the following tasks only if you upgraded your database from release 7.3.4. These tasks are *not* required if you upgraded your database from release 8.0.6 or higher.

Rebuild Unusable Bitmap Indexes

During the upgrade, 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 Tuning 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 Oracle9i, 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

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.

Upgrade to the New Release of Oracle Net Services (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: *Oracle9i Net Services Administrator's Guide* for more information about the advantages of Oracle Net, and see [Appendix B, "Upgrade Considerations 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 new Oracle9i database using the testing plan you developed in "[Develop a Testing Plan](#)" on page 2-8. 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 new Oracle9i database should be as good as, or better than, the performance of the previous database. If you notice any decline in database performance with the new Oracle9i database, then make sure the initialization parameters are set properly, because improperly set initialization parameters can impede performance.

Tune the Upgraded Database

If you want to improve the performance of the upgraded database, then tune the database. Actions you used to tune your previous database and applications should not impair the performance of the upgraded Oracle9i database.

See Also: *Oracle9i Database Performance Tuning Guide and Reference* for tuning information

Changing the Word Size of Your Current Release

The instructions in this section guide you through the process of changing the word size of your current release (switching from 32-bit software to 64-bit software or from 64-bit software to 32-bit software).

See Also: "[Changing Word Size](#)" on page 1-11 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 a user with SYSDBA privileges.

3. Run SHUTDOWN IMMEDIATE on the database:

```
SQL> SHUTDOWN IMMEDIATE
```

Note: For Oracle9i Real Application Clusters, issue this statement for all instances. Also, set the `CLUSTER_DATABASE` initialization parameter to `false`. You can change it back to `true` after the change in word size 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 Universal 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 release. Or, if you currently have a 64-bit installation, then install the 32-bit release.

Note: Installation and deinstallation are operating system-specific. For installation and deinstallation instructions, see your Oracle9i operating system-specific installation documentation and the Oracle9i README for your operating system.

7. Copy configuration files to a location outside of the old Oracle home:

- a. If your 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 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 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 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 new Oracle9i Oracle home. The name and location of the password file are operating system-specific; for example, on UNIX platforms, the default password file is *ORACLE_HOME/dbs/orapwsid*, but on Windows operating systems, the default password file is *ORACLE_HOME\database\pwsid.ora*. In both cases, *sid* is your Oracle instance ID.

Note: For Oracle9i Real Application Clusters, perform this step on all nodes. Also, if your *initdb_name.ora* file resides within the old environment's Oracle home, then move or copy the *initdb_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 a user with SYSDBA privileges.

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` command:

```
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 the `STANDARD` and `DBMS_STANDARD` packages, 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 issued a `SET ECHO ON` command, then you may want to issue a `SET ECHO OFF` command 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.

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 a COMPATIBLE Setting](#)
- [What Is Interoperability?](#)
- [Compatibility and Interoperability Issues Between Release 9.2 and Release 9.0.1](#)
- [Compatibility and Interoperability Issues Between Release 9.2 and Previous Releases](#)

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 parameter file, it defaults to the lowest possible setting for the release, which is `8.1.0` for all Oracle9i releases. You cannot use new features that would make your database incompatible until you raise the value of the `COMPATIBLE` initialization parameter.

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 a COMPATIBLE Setting"](#) on page 5-10 for a list of features that require the `COMPATIBLE` initialization parameter

Depending on the products you chose to install during your installation of the new Oracle9i release, the Oracle Universal Installer may set the `COMPATIBLE` initialization parameter to a higher value, such as `9.2.0`. Check your parameter file if you are unsure of the current setting of the `COMPATIBLE` initialization parameter.

Figure 5-1 illustrates the default settings and the possible settings of the COMPATIBLE initialization parameter in release 8.0, release 8.1, release 9.0, and release 9.2.

Figure 5-1 The COMPATIBLE Initialization parameter

Default 8.0.0	Default 8.0.0	Default 8.1.0	Default 8.1.0
 Release 8.0	 Release 8.1	 Release 9.0	 Release 9.2
Can be set to 8.0.x only	Can be set to 8.1.y or 8.0.x	Can be set to 9.0.y or 8.1.x	Can be set to 9.2.z, 9.0.y, or 8.1.:
Lowest Possible Setting: 8.0.0	Lowest Possible Setting: 8.0.0	Lowest Possible Setting: 8.1.0	Lowest Possible Setting: 8.1.0
Highest Possible Setting: Your current release	Highest Possible Setting: Your current release	Highest Possible Setting: Your current release	Highest Possible Setting: Your current release
Cannot be set to: <ul style="list-style-type: none"> • Any Oracle7 release or lower • Any release higher than current, including 8.1.0 or higher 	Cannot be set to: <ul style="list-style-type: none"> • Any Oracle7 release or lower • Any release higher than current, including 9.0.0 or higher 	Cannot be set to: <ul style="list-style-type: none"> • Any 8.0 release or lower • Any release higher than current, including 9.2.0 or higher 	Cannot be set to: <ul style="list-style-type: none"> • Any 8.0 release or • Any release higher than current, includ 10.0.0 or higher

How the COMPATIBLE Initialization Parameter Operates

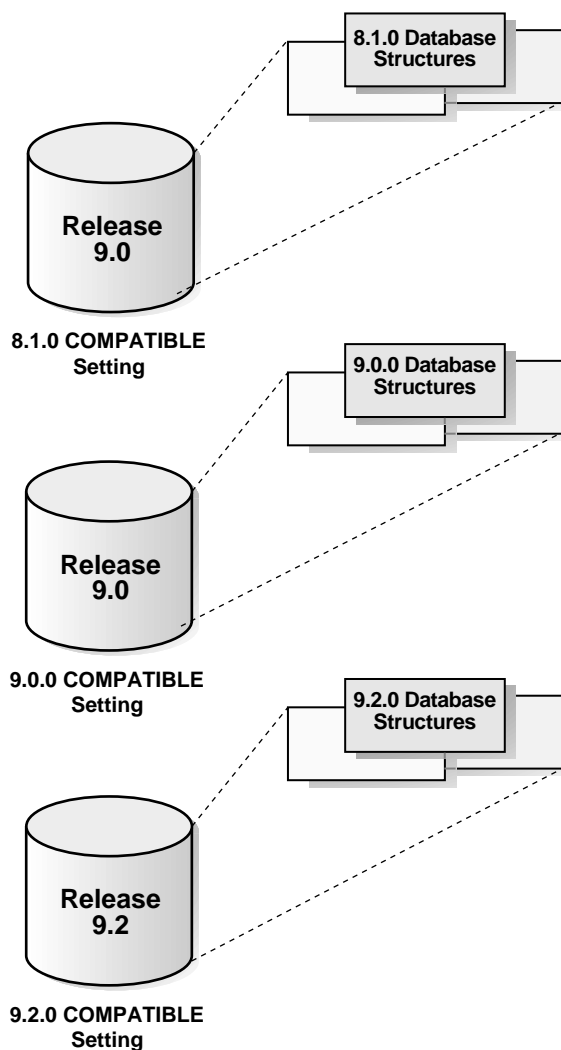
The COMPATIBLE initialization parameter operates in the following way:

- It controls the behavior of your database. For example, if you run a release 9.2 database with the COMPATIBLE initialization parameter set to 8.1.6, then the release 9.2 database generates release 8.1.6 compatible database structures on disk. Therefore, the COMPATIBLE initialization parameter 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, 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 to an appropriate value for the database.

Figure 5–2 Database Structures Depend on the COMPATIBLE Setting



See Also: *Oracle9i Database Concepts* for more information about database structures

Downgrading and Compatibility

Once you upgrade 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, or impossible, 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 7, "Downgrading a Database Back to the Previous Oracle Release"](#) for more information about downgrading

Compatibility Level

The compatibility level of your database corresponds to the value of the `COMPATIBLE` initialization parameter. For example, if you set the `COMPATIBLE` initialization parameter to `8.1.6`, then the database runs at 8.1.6 compatibility level.

Checking the Current Value of the `COMPATIBLE` Initialization Parameter

To check the current value of the `COMPATIBLE` initialization parameter, 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 during the 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: Once the upgrade or downgrade is complete, you can change the setting of the `COMPATIBLE` initialization parameter as necessary.

Setting the `COMPATIBLE` Initialization Parameter

Complete the steps in one of the following sections to set the `COMPATIBLE` initialization parameter:

- [Raising the `COMPATIBLE` Initialization Parameter](#)
- [Lowering the `COMPATIBLE` Initialization Parameter](#)

Raising the `COMPATIBLE` Initialization Parameter

Complete the following steps to set the `COMPATIBLE` initialization parameter to a higher value:

1. Perform a backup of your database before you raise the `COMPATIBLE` initialization parameter (optional).

Raising 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 Backup and Recovery Concepts* for more information about performing a backup

2. If you are using a server parameter file, then complete the following steps:
 - a. Update the server parameter file to set or change the value of the `COMPATIBLE` initialization parameter.

For example, to set the `COMPATIBLE` initialization parameter to 9.2.0, issue the following statement:

```
SQL> ALTER SYSTEM SET COMPATIBLE = '9.2.0' SCOPE=SPFILE;
```

- b. Shut down and restart the instance.
3. If you are using an initialization parameter file, then complete the following steps:
 - a. Shut down the instance if it is running:

```
SQL> SHUTDOWN IMMEDIATE
```

- b. Edit the initialization parameter file to set or change the value of the `COMPATIBLE` initialization parameter.

For example, to set the `COMPATIBLE` initialization parameter to 9.2.0, enter the following in the initialization parameter file:

```
COMPATIBLE = 9.2.0
```

- c. Start the instance using `STARTUP`.

Lowering the `COMPATIBLE` Initialization Parameter

Complete the following steps to set the `COMPATIBLE` initialization parameter to a lower value:

1. Make sure that your database does not have any incompatibilities with the intended lower value of the `COMPATIBLE` initialization parameter.

See Also: ["Remove Incompatibilities"](#) on page 7-2 for information on removing incompatibilities

2. If you are using any initialization parameters that were added in a release higher than the intended lower value of the `COMPATIBLE` initialization parameter, then remove them from your parameter file.

See Also: The "What's New in Oracle9i Database Reference" section of *Oracle9i Database Reference* for lists of initialization parameters added in each Oracle9i release

3. Issue an `ALTER DATABASE RESET COMPATIBILITY` statement:

```
SQL> ALTER DATABASE RESET COMPATIBILITY;
```

See Also: ["About ALTER DATABASE RESET COMPATIBILITY"](#) on page 5-9 for more information

4. If you are using a server parameter file, then complete the following steps:
 - a. Update the server parameter file to set or change the value of the `COMPATIBLE` initialization parameter.

For example, to set the `COMPATIBLE` initialization parameter to 9.0.0, issue the following statement:

```
SQL> ALTER SYSTEM SET COMPATIBLE = '9.0.0' SCOPE=SPFILE;
```

- b. Shut down and restart the instance.
5. If you are using an initialization parameter file, then complete the following steps:

- a. Shut down the instance if it is running:

```
SQL> SHUTDOWN IMMEDIATE
```

- b. Edit the initialization parameter file to set or change the value of the `COMPATIBLE` initialization parameter.

For example, to set the `COMPATIBLE` initialization parameter to `9.0.0`, enter the following in the initialization parameter file:

```
COMPATIBLE = 9.0.0
```

- c. Start the instance using `STARTUP`.

About ALTER DATABASE RESET COMPATIBILITY

You use the `ALTER DATABASE RESET COMPATIBILITY` statement to instruct Oracle that you want to lower the compatibility level of your database. Some Oracle features, such as undo tablespaces, 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 undo tablespace, then the undo tablespace is a 9.0.0 compatible object in the database.

`ALTER DATABASE RESET COMPATIBILITY` checks each feature that may have created an object that is incompatible with the lowest possible compatibility level, which is 8.1.0 for all Oracle9i releases. If the check indicates that no incompatible objects exist for a certain feature, then the compatibility level of that 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 lowest possible compatibility level that enables the feature.

For example, if one or more undo tablespaces exist, then the compatibility level for the undo tablespaces feature is set to 9.0.0, because 9.0.0 is the lowest possible compatibility level that enables the undo tablespaces 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 a 9.0.0 or higher compatibility level.

If you close the database, lower the value of the `COMPATIBLE` initialization parameter, and then open the database, Oracle checks the compatibility level of each feature. If a feature has a compatibility level higher than the value of the `COMPATIBLE` initialization parameter, then the database fails to open and displays an error message indicating the incompatible features.

If you remove all of the incompatibilities that exist in your database, but fail to issue 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. 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 a COMPATIBLE Setting

To use the features listed in [Table 5-1](#), the `COMPATIBLE` initialization parameter must be set to the indicated value. The features listed do *not* represent a complete list of Oracle features. Instead, the features listed are only those Oracle features that require a compatibility level; some features do not require a compatibility level.

See Also:

- *Oracle9i Database New Features* for more information about the features listed in the following sections and for information about other new release 9.2 features
- *Oracle9i Database Master Index* for entries relating to the new release 9.2 features

Table 5–1 Features Requiring A COMPATIBLE Setting

Feature Identifier	Compatibility Level	Description
DEFPART	9.2.0.0.0	Release 9.2 DEFAULT partitions: <ul style="list-style-type: none"> ▪ DEFAULT partitions on list partitioned tables
MV92	9.2.0.0.0	Release 9.2 materialized views: <ul style="list-style-type: none"> ▪
PARTMCLS	9.2.0.0.0	Release 9.2 partitioning methods: <ul style="list-style-type: none"> ▪ Partitioning of tables using range-list methods
KNL92	9.2.0.0.0	Release 9.2 Streams
SPTEMPL	9.2.0.0.0	Release 9.2 subpartition templates <ul style="list-style-type: none"> ▪ Subpartition templates in composite partitioned tables
FGASYNPL	9.2.0.0.0	Fine-grained security synonym policy
HSC	9.2.0.0.0	Heap segment block compression
LOB_RET	9.2.0.0.0	LOB retention: <ul style="list-style-type: none"> ▪ Retention stored in LOB columns
LMST	9.2.0.0.0	Locally managed SYSTEM tablespace
COLLOCT	9.2.0.0.0	Ordered collections in tables <ul style="list-style-type: none"> ▪ Ordered collections stored in tables
PGTMGDLB	9.2.0.0.0	Automatic segment-space managed tablespaces with LOBs: <ul style="list-style-type: none"> ▪ LOB columns in automatic segment-space managed tablespaces
RLENG	9.2.0.0.0	Rules engine
SYNUDC	9.2.0.0.0	Type synonyms or user-defined constructors
XMLSBSTR	9.2.0.0.0	XMLSchema based XMLType storage
PGTMGDTS	9.0.1.3.0	Automatic segment-space managed tablespaces
MV90	9.0.0.0.0	Release 9.0 materialized views
PARTM82	9.0.0.0.0	Release 9.0 partitioning methods: <ul style="list-style-type: none"> ▪ Partitioning of tables using list methods
APPROLE	9.0.0.0.0	Application role
LGMR_B	9.0.0.0.0	Basic LogMiner info

Table 5–1 (Cont.) Features Requiring A COMPATIBLE Setting

Feature Identifier	Compatibility Level	Description
CPTLEN	9.0.0.0.0	Code point length
EXTTAB	9.0.0.0.0	Create external tables
WRDIR	9.0.0.0.0	Directory write privilege
DOMINDEA	9.0.0.0.0	Domain indexes on embedded ADTs
DOMINIOT	9.0.0.0.0	Domain indexes on index-organized tables
DOMINDRM	9.0.0.0.0	Domain indexes with row movement
EJTYPE	9.0.0.0.0	External Java types
APFFGA	9.0.0.0.0	Fine-grained auditing
LGMR_F	9.0.0.0.0	Full LogMiner info
FDOMIND	9.0.0.0.0	Function-based domain indexes
HASHPIOT	9.0.0.0.0	Hash partitioned index-organized tables
IOTBULOG	9.0.0.0.0	Index-organized tables batch update logging
IOTCVLOG	9.0.0.0.0	Index-organized tables column vector logging
IOTWMAP	9.0.0.0.0	Index-organized tables with mapping tables
URIDIND	9.0.0.0.0	Indexes on UROWIDs
JOININD	9.0.0.0.0	Join indexes
LGINDKEY	9.0.0.0.0	Large index keys
LDOMIND	9.0.0.0.0	Local domain indexes
LOGSTDBY	9.0.0.0.0	Logical standby
MLCTABLE	9.0.0.0.0	Multi level collection in tables
MULTBZ	9.0.0.0.0	Multiple block sizes
NFSTABLE	9.0.0.0.0	Not final type or subtype in tables
PDMLITLS	9.0.0.0.0	PDML ITL invariants
PIOTLOBS	9.0.0.0.0	Partitioned index-organized tables with LOBs: <ul style="list-style-type: none"> ■ LOB columns in partitioned index-organized tables ■ Varray columns in partitioned index-organized tables
TXNAUDN	9.0.0.0.0	Redo for transaction name auditing

Table 5–1 (Cont.) Features Requiring A COMPATIBLE Setting

Feature Identifier	Compatibility Level	Description
NESTEDTX	9.0.0.0.0	Redo/undo for nested transactions
ROWDEP	9.0.0.0.0	Row level dependencies
STAUTOFM	9.0.0.0.0	Standby automatic file management
TYPEVL	9.0.0.0.0	Type evolution
UNDOTBSP	9.0.0.0.0	Undo tablespaces
VWCONSTR	9.0.0.0.0	View constraints
ANYTABLE	9.0.0.0.0	XMLType/AnyType/AnyData in tables
ALTERFRL	8.1.6.0.0	Alter freelists: <ul style="list-style-type: none"> ■ Change FREELIST specification in ALTER statements
CARELOB	8.1.6.0.0	Cache reads mode for LOBs
EDTRIG	8.1.6.0.0	Enhanced DDL/DML support in triggers
FASTDROP	8.1.6.0.0	Faster segment drop
OPQTYPE	8.1.6.0.0	Opaque types
TBSMIGTN	8.1.6.0.0	Tablespace migration
TBSTRNSG	8.1.6.0.0	Transient segments

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 a release 7.3.4 database to understand the object SQL extensions that are available only with release 8.0 and higher.

Note: Since this book documents 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 Between Release 9.2 and Release 9.0.1

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 release 9.2 and release 9.0.1:

- [Locally Managed SYSTEM Tablespace](#)
- [Dictionary Managed Tablespaces](#)
- [Change in Compatibility for Automatic Segment-Space Managed Tablespaces](#)
- [Compatibility and Object Types](#)
- [Oracle Managed Files](#)
- [Oracle OLAP](#)
- [Log Format Change with Parallel Redo](#)
- [Oracle Dynamic Services](#)
- [Oracle Syndication Server](#)

Locally Managed SYSTEM Tablespace

The SYSTEM tablespace can be locally managed only if COMPATIBLE is set to 9.2.0 or higher. The SYSTEM tablespace can be migrated from dictionary managed format to locally managed format using the DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL procedure.

Before the SYSTEM tablespace can be migrated to locally managed format, you should ensure the following:

- The database has a default temporary tablespace which is not SYSTEM

- There are not any rollback segments in dictionary managed tablespaces
- There is at least one online rollback segment in a locally managed tablespace, or an undo tablespace (if using automatic undo management mode) should be online.
- All tablespaces other than the tablespace containing the undo space (undo tablespace or the tablespace containing the rollback segment) and the default temporary tablespace are in read-only mode.
- There is a cold backup of the system.
- The system is in restricted mode.

Once the `SYSTEM` tablespace has been migrated to locally managed format, you will not be able to downgrade back to a previous release. The following query determines whether the `SYSTEM` tablespace is locally managed:

```
SQL> SELECT ts# FROM ts$  
       WHERE ts# = 0 AND bitmapped <> 0;
```

If 0 rows is returned, then the `SYSTEM` tablespace is dictionary managed. Otherwise, the `SYSTEM` tablespace is locally managed.

Dictionary Managed Tablespaces

Starting with release 9.2, dictionary managed tablespaces are deprecated. Once the `SYSTEM` tablespace has been migrated from dictionary managed format to locally managed format, existing dictionary managed tablespaces are read-only. That is, they cannot be made read-write once the `SYSTEM` tablespace is locally managed.

Once the `SYSTEM` tablespace is locally managed (either due to a new installation of `SYSTEM` tablespace migration), new dictionary managed tablespaces cannot be created.

Change in Compatibility for Automatic Segment-Space Managed Tablespaces

Starting with release 9.0.1.3.0, the compatibility requirement for automatic segment-space managed tablespaces has been changed from 9.0.0.0.0 when first introduced in release 9.0.1.0.0 to 9.0.1.3.0. If you are upgrading from an Oracle9i release earlier than release 9.0.1.3.0 and the database contains any automatic segment-space managed tablespaces, then the `COMPATIBLE` initialization parameter will need to be set to 9.0.1.3.0 or higher in order to open the database. The existing tablespaces need not be dropped.

Once the database has been opened with `COMPATIBLE` set to 9.0.1.3.0 or higher, it can only be downgraded to release 9.0.1.3.0 or higher if automatic segment-space managed tablespaces are used.

Compatibility and Object Types

Starting with release 9.2, object types support user-defined constructors using the `CONSTRUCTOR` keyword that cannot be referred to from PL/SQL programs in previous releases of Oracle. Specifically, such programs will fail to compile with an error.

Oracle Managed Files

Starting with release 9.0.1.2.0, the naming scheme used by Oracle to keep track of Oracle Managed Files has changed. As a result, existing Oracle Managed Files created in Oracle9i releases earlier than release 9.0.1.2.0 will appear to Oracle to be regular operating system files. See [Table , "Migrate Your Oracle Managed Files"](#) on page 4-2 for information on migrating your Oracle Managed Files to the new naming scheme.

Oracle OLAP

the OLAP API client provided with release 9.0.1 is not compatible with later Oracle releases; similarly, the OLAP API client provided with release 9.2 is not compatible with earlier Oracle releases.

The procedure that an application uses to make a connection through the OLAP API has changed in release 9.2. Connections in previous releases relied on CORBA software, but in release 9.2, connections are made through Java Database Connectivity (JDBC). Consequently, programs created using the OLAP API client provided with release 9.0.1 will not execute in later releases, and programs created using the OLAP API client provided with release 9.2 will not execute in earlier Oracle releases.

To upgrade OLAP API applications designed to run in release 9.0.1, application developers must use the OLAP API client provided with release 9.2 and revise the code for making a connection and for creating a `MetadataProvider`.

For information about using the OLAP API in release 9.2 to perform these actions, see the *Oracle9i OLAP Developer's Guide to the OLAP API* and the online Oracle OLAP API Reference help provided with release 9.2.

Log Format Change with Parallel Redo

Starting with release 9.2, the parallel redo feature generates redo logs using a new format. Previous releases of Oracle cannot apply parallel redo generated logs. However, when Oracle9i release 9.0.1 detects that release 9.2 parallel redo is being applied, the following error is displayed:

```
ORA-00303: cannot process Parallel Redo
```

The new log format requires a clean shutdown of the database before a downgrade. A complete backup is also highly recommended. If an ORA-00303 error is displayed after the downgrade, then you must upgrade to release 9.2, perform recovery, shut down the database cleanly, and then perform the downgrade again.

Release 9.2 can process release 9.0.1 and earlier format logs as well as release 9.2 parallel redo format logs.

Oracle Dynamic Services

Starting with release 9.2, Oracle Dynamic Services has been Deprecated. Oracle Dynamic Services, an XML-based broker for the creation, aggregation, and deployment of services from various content sources, was released with Oracle9i Database release 9.0.1 along with the documentation, *Oracle Dynamic Services User's and Administrator's Guide*.

Starting with Oracle9iAS release 2 (9.0.2), Oracle Corporation is delivering an integrated, J2EE-compliant Web Services platform. Oracle Dynamic Services has been integrated with Oracle9iAS Web Services as the XML/HTML Stream Processing Tool.

See Also: *Oracle9i Application Server Web Services Developer's Guide* for more information

Oracle9iAS release 2 (9.0.2) provides a standards-based, fully integrated J2EE and Web services deployment platform. The current Dynamic Services functionality has been integrated into the Oracle9iAS platform, and the Dynamic Services terminal release is being delivered with Oracle9i Database release 9.2.

Oracle Syndication Server

Starting with release 9.2, Oracle Syndication Server has been Deprecated. Oracle Syndication Server, designed to deliver file system and database content to Information and Content Exchange (ICE)-compliant subscribers, was released with

Oracle9i Database release 9.0.1 along with the documentation, *Oracle Syndication Server User's and Administrator's Guide*.

Starting with Oracle9iAS release 2 (9.0.2), Oracle Syndication Server has become a feature of Oracle9iAS. The current Syndication Server functionality has been integrated into this platform, and the Syndication Server terminal release is being delivered with Oracle9i Database release 9.2.

Oracle9iAS Syndication Server is automatically installed with the Oracle9iAS Portal install. The current release of the *Oracle Syndication Server User's and Administrator's Guide* can be found with the Oracle9iAS Portal documentation on the Oracle9iAS release 2 (9.0.2) Documentation CD-ROM.

Compatibility and Interoperability Issues Between Release 9.2 and Previous Releases

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](#)
- [Miscellaneous Compatibility and Interoperability Issues](#)

Applications

You do not need to modify existing applications that do not use new release 9.2 features. Existing applications should achieve the same, or enhanced, functionality on release 9.2. To increase the likelihood that applications running against your release 9.2 database will continue to work if you downgrade to a previous release, you can set the `COMPATIBLE` initialization parameter to match the previous release.

However, the `COMPATIBLE` initialization parameter only restricts the use of release 9.2 features that change the formatting on disk, not the use of other release 9.2 features. Therefore, a setting lower than 9.2.0 does not guarantee that applications developed in release 9.2 will run correctly if the database is downgraded to a previous release.

See Also: [Chapter 6, "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 OLOG 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 Oracle9i link line differs from the Oracle7 link line. See the `ORACLE_HOME/rdbms/demo/demo_rdbms.mk` file for examples of using the Oracle9i link line as an Oracle9i OCI application is compiled.

Oracle7 Clients Oracle7 clients can make selective use of Oracle9i OCI, combining Oracle7 and Oracle9i 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 Oracle9i 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 Oracle9i calls if they want to combine Oracle7 code with Oracle9i 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 Oracle9i, 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 Oracle9i:

```
ALTER SESSION SET events = '10933 trace name context forever, level 512';
```

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.

If you are upgrading from release 8.1.7 and this event exists in your parameter file, then, as a temporary workaround, change all occurrences of this event from `event = '10933 trace name context forever, level 512'` to `event = '10933 trace name context forever, level 1024'`.

Default Value of Parameter for Functions or Procedures in the Spec and Body Do Not Match In previous releases, PL/SQL quietly ignored this error and used the default value specified in the spec (ignoring the possibly different value in the body). Also, if there is no default value specified in the spec, and a default value is specified in the body, then the default value in the body is ignored.

In Oracle9i, PL/SQL will flag such discrepancies as errors. It is recommended to fix the code, if such errors are reported, to avoid any possible future bugs.

If you are unable to immediately modify the PL/SQL code, then Oracle provides an event to temporarily restore the old compiler behavior:

```
ALTER SESSION SET events = '10932 trace name context level 32768'
```

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 Oracle9i, 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 5–2 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 Oracle9i, 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/initSID.ora`, where `SID` is the instance name.

In Oracle9i, 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/spfileSID.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/initSID.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.

CREATE TABLESPACE: New Behavior

In Oracle8i, the default type of tablespace that is created is dictionary managed if the `EXTENT MANAGEMENT` clause is not specified in the `CREATE TABLESPACE` statement.

In Oracle9i, the default for the `EXTENT MANAGEMENT` clause depends on the setting of the `COMPATIBLE` initialization parameter:

- If `COMPATIBLE` is set to 8.1.x, then the Oracle8i behavior is preserved. That is, the tablespace is created as dictionary managed.
- If `COMPATIBLE` is set to 9.0.0 or higher, then the default is locally managed. The default storage clause is parsed to determine whether to use `AUTOALLOCATE` or `UNIFORM` allocation policy for this tablespace.

In addition, there was another change made to disallow assigning permanent locally managed tablespaces as a user's temporary tablespace. In Oracle8i, an error would be signalled only when a temporary segment had to be created in the tablespace.

Default Temporary Tablespaces

Oracle Corporation strongly recommends using a default temporary tablespace for the database. In a future release, it will be mandatory to have one. The default temporary tablespace should be created using the `CREATE TEMPORARY TABLESPACE` statement.

Undo 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

Data Dictionary

This section describes possible compatibility and interoperability issues resulting from data dictionary changes.

See Also: [Appendix A, "Changes to Initialization Parameters and the Data Dictionary"](#) for more information about obsolete and deprecated dictionary 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 Oracle9i, 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.

Oracle9i eliminates these unintentional invalidations.

Datatypes

This section describes compatibility and interoperability issues relating to datatypes.

Datetime and Interval Datatypes

When a database is upgraded to Oracle9i, 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 upgrade from Oracle7 to Oracle9i, then the index data was relocated automatically during migration. However, the index data was not relocated if you used the `MIG` utility or the Database Upgrade 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: ["Migration Issues for Physical Rowids"](#) on page D-32 for more information about the new physical rowid format

AL24UTFFSS Character Set Desupported

The AL24UTFFSS Unicode character set has been desupported in Oracle9i. 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 Oracle9i, 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 Oracle9i. 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 Database Globalization Support Guide* for more information about the Character Set Scanner

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 *xS* 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 Oracle9i. 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 or Higher Server Any transfer involving data of non-final types will return an error. Release 8.1 clients cannot access a release 9.0 or higher server if the type has been altered to non-final on the server.

Release 9.0 and Higher 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. 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 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 or Higher User-Defined Datatypes When a release 8.0 client accesses a server with release 8.1 or higher 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 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 Oracle9i 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 Oracle9i, you must always run the `utlchn1.sql` script.

SQL Scripts `utlexcpt.sql` and `utlexpt1.sql`

The Oracle9i 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 Oracle9i, 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.

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

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

Oracle9i contains a significant number of optimizer enhancements that are either new or have not been enabled by default in previous releases.

Upgrading an existing application to Oracle9i could therefore result in a large number of changes in execution plans. For a mature application, changes in

behavior may introduce an element of risk. Customers who wish to minimize execution plan changes can do so by means of the `OPTIMIZER_FEATURES_ENABLE` initialization parameter.

Setting the value of this parameter to an earlier release, for example, release 8.1.7, makes Oracle use only those optimizer features that were enabled by default in that release, something that will reduce the likelihood of changes in execution plans when upgrading from that release.

The Oracle Plan Stability feature can also be used to preserve old behavior when upgrading to a new release.

Oracle9i Real Application Clusters

Support for different releases of Oracle within one Oracle9i 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 Oracle9i Real Application Clusters environment is supported on your operating system.

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 5–3](#) 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 5–3](#) 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 5–3 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

Table 5–3 (Cont.) Conversion of INSTANCES Keyword in Release 8.1

Degree	Instances	8.1 Degree
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 Oracle8i; 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 4-18 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 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 later Oracle release, with the following exception: a backup of an Oracle7 database taken after running the `MIG` utility can be restored and recovered with Oracle9*i*. If EBU is used to backup the Oracle7 database, and the database must later be restored for recovery with Oracle9*i*, then you must use EBU to restore the datafiles prior to recovering them with Oracle9*i*. 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 release 8.0 or higher can be restored and recovered with any later Oracle release, if a direct upgrade path between the release that backed-up the file and the release that recovers the file is supported in [Table 2-1, "Upgrade Paths"](#) on page 2-3. You can also restore and recover release 8.0 and higher backups with an earlier release 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. Oracle9i 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_REOBJECT`
- `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 release 8.0 and higher. 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 *Oracle9i 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 *Oracle9i 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: *Oracle9i Net Services Administrator's Guide* for more information about using connection load balancing and the SERVICE_NAME parameter.

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.

Upgrading Your Applications

This chapter describes upgrading your current applications and covers the following topics:

- [Overview of Upgrading Applications](#)
- [Upgrading Precompiler and OCI Applications](#)
- [Upgrading SQL*Plus Scripts](#)
- [Upgrading Oracle7 Forms or Oracle Developer Applications](#)

Overview of Upgrading Applications

You do not need to modify existing applications that do not use features available in the new Oracle9i release. Existing applications running against a new Oracle9i 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 upgrading to the new Oracle9i release. 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 the features available in the new Oracle9i release

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 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 releases or the application of certain patch sets.

See Also:

- ["Applications"](#) on page 5-19 for information about compatibility issues that relate to applications
- [Appendix A, "Changes to Initialization Parameters and the Data Dictionary"](#) for information about initialization parameter changes and data dictionary changes
- *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 release 2.x, Net8, and Oracle Net Services work with various Oracle releases. Thus, Oracle7, Oracle8, Oracle8*i*, and Oracle9*i* databases can communicate by using SQL*Net release 2.x, Net8, and Oracle Net Services. SQL*Net release 1.x, 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 new 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

Two types of upgrades are possible for both client and server Oracle software.

Major Database Release Upgrade The upgrade changes the first digit of the release number. For example, upgrading from Oracle8*i* to Oracle9*i* is a major database release upgrade.

Database Maintenance Release Upgrade The upgrade changes the second digit of the release number. For example, upgrading from release 9.0.1 to release 9.2 is a database maintenance release upgrade.

Note: Starting with release 9.2, maintenance releases of Oracle are denoted by a change to the second digit of a release number. In previous releases, the third digit indicated a particular maintenance release.

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 6-3.

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 6-6.

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 8.1.7, then your Oracle server software *should* be release 8.1.7 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 and higher client software can run against an Oracle7 server. 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 9.2, then Oracle Corporation recommends upgrading the client software to release 9.2 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.

Dynamically-Linked Applications Do Not Need To Be Relinked When you perform an 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 Oracle9i 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 features that are available in the new Oracle9*i* release. In addition, your application cannot leverage some of the possible performance benefits of the new Oracle9*i* release.

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 or Compile the Application Using the New Software

You can precompile or compile the application with the new Oracle9i 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 Oracle9i Features

You can make code changes to your application to take advantage of new Oracle9i 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 Oracle9i. Further, you can add new features to your application that are available only with the new release of Oracle9i.

Become familiar with the new Oracle9i 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 Oracle9i 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 Oracle9i.
 - If you have a version 8 application and you want to take advantage of the new Oracle9i features, then incorporate code for the new Oracle9i 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 Oracle9i runtime library, SQLLIB, which is included with the precompiler.

Changing OCI Applications Complete the following steps to change your OCI application to use Oracle9i 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 Oracle9i OCI calls.
 - If your application uses Oracle7 OCI calls, then incorporate Oracle9i 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 Oracle9i OCI calls into the existing application.
2. Compile the application.
3. Relink the application with the Oracle9i 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 Oracle9i, 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 Oracle9i functionality, change existing SQL scripts to use the new Oracle9i syntax. Existing SQL scripts run unchanged on Oracle9i, and require no modification, if they do not use new Oracle9i 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 Oracle9i. However, review the new features described in *Oracle9i Database New Features* to determine whether any of the new Oracle9i features would be beneficial to your applications or might otherwise affect them. Information about the ways in which the Oracle9i 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.

Downgrading a Database Back to the Previous Oracle Release

This chapter guides you through the process of downgrading a database back to the previous Oracle release. 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 instructions about downgrading on your operating system.

Perform a Full Offline Backup

Perform a full offline backup of your release 9.2 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 is determined by 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 9.0.1 and the `COMPATIBLE` parameter is set to 9.0.0 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 see "[Downgrade the Database](#)" on page 7-21.

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.

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. 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.

6. Query the `V$COMPATIBILITY` dynamic performance view to identify any 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 are not currently in use, and no action is required for them.

7. Run `utlincmp.sql`:

```
SQL> SPOOL utlincmp.log
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.log` file and look for instances where a `SELECT` statement returned values. The values returned are incompatibilities that may need to be removed depending on the release to which you are downgrading.

8. 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 Oracle releases. Depending on the release to which you are downgrading, you may need to complete the steps in some or all of the following sections.

For example, if you are downgrading to release 9.0.1, then you only need to complete the steps in "[Removing Release 9.2 Incompatibilities](#)" on page 7-4. However, if you are downgrading to release 8.1.7, then you need to complete the steps in "[Removing Release 9.2 Incompatibilities](#)" on page 7-4 as well as the steps in "[Removing Release 9.0.1 Incompatibilities](#)" on page 7-8.

Note: If you are downgrading from Oracle9i Enterprise Edition to Oracle9i (formerly Workgroup Server), then, before you downgrade, modify any applications that use the advanced features of Oracle9i 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 Release 9.2 Incompatibilities

If you are downgrading to release 9.0.1 or lower, then complete the actions in the following sections to remove incompatibilities:

- [Release 9.2 DEFAULT Partitions](#)
- [Release 9.2 Partitioning Methods](#)
- [Release 9.2 Streams](#)
- [Release 9.2 Subpartition Templates](#)
- [LOB Retention](#)
- [Automatic Segment-Space Managed Tablespaces with LOBs](#)

Release 9.2 DEFAULT Partitions

This section describes removing incompatibilities relating to release 9.2 DEFAULT partitions.

Drop All DEFAULT Partitions on List Partitioned Tables Before you downgrade to release 9.0.1 or lower, drop all DEFAULT partitions on list partitioned tables. To identify all list partitioned tables with DEFAULT partitions, issue the following SQL statement:

```
SELECT u.name AS OWNER, o.name AS TABLE_NAME, o.subname AS PARTITION_NAME
FROM sys.user$ u, sys.obj$ o, sys.tabpart$ tp
WHERE BITAND(tp.flags, 16384) = 16384
AND tp.obj# = o.obj# AND o.owner# = u.user#;
```

For each partition represented by the `PARTITION_NAME` column in the table represented by the `OWNER.TABLE_NAME` columns, simply drop the partition:

```
ALTER TABLE OWNER.TABLE_NAME DROP PARTITION PARTITION_NAME;
```

Release 9.2 Partitioning Methods

This section describes removing incompatibilities relating to release 9.2 partitioning methods.

Drop All Partitioned Tables That Use Range-List Methods Before you downgrade to release 9.0.1 or lower, drop all partitioned tables that use range-list methods. To identify existing tables partitioned with range-list methods, issue the following SQL statement:

```
SELECT u.name AS OWNER, o.name AS TABLE_NAME
FROM sys.user$ u, sys.obj$ o, sys.partobj$ po
WHERE po.parttype = 1 AND MOD(po.spare2, 256) = 4
AND o.obj# = po.obj# AND o.owner# = u.user#;
```

If you do not need to preserve the table data, then, for each table represented by the `OWNER.TABLE_NAME` columns, simply drop the table:

```
DROP TABLE OWNER.TABLE_NAME;
```

However, if you need to preserve the table data, then copy the data into non-partitioned tables, or copy the data into tables partitioned by range, hash, list, or another composite method.

Release 9.2 Streams

This section describes removing incompatibilities relating to release 9.2 Streams.

Drop All Streams Capture Processes Before you downgrade to release 9.0.1 or lower, drop all Streams capture processes. To identify existing capture processes, issue the following SQL statement:

```
SELECT capture_name FROM dba_capture;
```

For each capture process listed in the `CAPTURE_NAME` column, issue the following SQL statement:

```
EXECUTE dbms_capture_adm.drop_capture(capture_name => 'CAPTURE_NAME');
```

Drop All Streams Apply Processes Before you downgrade to release 9.0.1 or lower, drop all Streams apply processes. To identify existing apply processes, issue the following SQL statement:

```
SELECT apply_name FROM dba_apply;
```

For each apply process listed in the `APPLY_NAME` column, issue the following SQL statement:

```
EXECUTE dbms_apply_adm.drop_apply(apply_name => 'APPLY_NAME');
```

Release 9.2 Subpartition Templates

This section describes removing incompatibilities relating to release 9.2 subpartition templates.

Drop All Subpartition Templates in Composite Partitioned Tables Before you downgrade to release 9.0.1 or lower, drop all subpartition templates in composite partitioned tables. To identify existing composite partitioned tables with subpartition templates, issue the following SQL statement:

```
SELECT u.name AS OWNER, o.name AS TABLE_NAME
       FROM sys.user$ u, sys.obj$ o
       WHERE o.owner# = u.user#
             AND o.obj# in (SELECT DISTINCT bo# FROM defsubpart$)
UNION
SELECT u.name AS OWNER, o.name AS TABLE_NAME
       FROM sys.user$ u, sys.obj$ o
       WHERE o.owner# = u.user#
             AND o.obj# in (SELECT DISTINCT bo# from defsubpartlob$);
```

For each table represented by the `OWNER.TABLE_NAME` columns, simply drop the subpartition template:

```
ALTER TABLE OWNER.TABLE_NAME SET SUBPARTITION TEMPLATE ();
```


LOB Retention

This section describes removing incompatibilities relating to LOB retention.

Drop Retention Stored in LOB Columns Before you downgrade to release 9.0.1 or lower, drop retention stored in LOB columns. To identify existing LOB columns with retention, issue the following SQL statement:

```
SELECT u.name AS OWNER, o.name AS TABLE_NAME, c.name AS LOB_COL_NAME
       FROM sys.user$ u, sys.obj$ o, sys.col$ c, sys.lob$ l
       WHERE BITAND(l.flags, 64) = 64 AND l.obj# = o.obj#
              AND c.obj# = o.obj# AND c.col# = l.col#
              AND o.owner# = u.user#;
```

For each column represented by the LOB_COL_NAME column in the table represented by the OWNER.TABLE_NAME columns, simply drop the retention:

```
ALTER TABLE OWNER.TABLE_NAME MODIFY LOB(LOB_COL_NAME)
       (REBUILD FREEPOLLS);
```

Automatic Segment-Space Managed Tablespaces with LOBs

This section describes removing incompatibilities relating to automatic segment-space managed tablespaces with LOBs.

Drop LOB Columns in Automatic Segment-Space Managed Tablespaces Before you downgrade to release 9.0.1 or lower, drop all LOB columns in automatic segment-space managed tablespaces. To identify existing automatic segment-space managed tablespaces with LOB columns, issue the following SQL statement:

```
SELECT u.name AS OWNER, o.name AS TABLE_NAME, c.name AS LOB_COL_NAME
       FROM sys.lob$ l, sys.ts$ t, sys.user$ u, sys.obj$ o, sys.col$ c
       WHERE l.ts# = t.ts# AND
              (DECODE(BITAND(t.flags, 32), 32, 1, 0) = 1 AND t.online$ <> 3) AND
              o.owner# = u.user# AND l.obj# = o.obj# AND
              l.obj# = c.obj# AND l.col# = c.col#;
```

For each LOB segment listed, perform one of the following actions:

- The LOB columns can be moved to a tablespace that is not auto segment-space managed:

```
ALTER TABLE OWNER.TABLE_NAME MOVE LOB(LOB_COL_NAME) STORE AS
       (TABLESPACE manual segment space-managed tablespace);
```

- The table containing the LOB columns can be dropped:

```
DROP TABLE OWNER.TABLE_NAME;
```

Removing Release 9.0.1 Incompatibilities

If you are downgrading to release 8.1.7 or lower, then complete the actions in the following sections to remove incompatibilities:

- [Tablespaces](#)
- [Schema Objects](#)
- [Release 9.0 Partitioning Methods](#)
- [Hash Partitioned Index-Organized Tables](#)
- [PDML ITL Invariants](#)
- [Partitioned Index-Organized Tables with LOBs](#)
- [Datatypes](#)
- [User-Defined Datatypes](#)
- [SQL and PL/SQL](#)
- [Constraints and Triggers](#)

Tablespaces

This section describes removing incompatibilities relating to tablespaces that were introduced in release 9.0.1.

Drop All Automatic Segment-Space Managed Tablespaces Before you downgrade to release 8.1.7 or lower, drop 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';
```

For each tablespace represented by the TABLESPACE_NAME column, simply drop the tablespace:

```
DROP TABLESPACE TABLESPACE_NAME;
```

Drop All Undo Tablespaces Before you downgrade to release 8.1.7 or lower, drop all undo tablespaces. To identify existing undo tablespaces, issue the following SQL statement:

```
SELECT name AS TABLESPACE_NAME FROM sys.ts$
       WHERE BITAND(flags, 16) = 16 AND online$ <> 3;
```

For each tablespace represented by the TABLESPACE_NAME column, simply drop the tablespace:

```
DROP TABLESPACE TABLESPACE_NAME;
```

Schema Objects

This section describes removing incompatibilities relating to schema objects that were introduced in release 9.0.1.

Drop All 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 u.name AS OWNER, o.name AS TABLE_NAME
       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;
```

For each table represented by the OWNER.TABLE_NAME columns, simply drop the table:

```
DROP TABLE OWNER.TABLE_NAME;
```

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.

Release 9.0 Partitioning Methods

This section describes removing incompatibilities relating to release 9.0 partitioning methods.

Drop All Partitioned Tables That Use List Methods Before you downgrade to release 8.1.7 or lower, drop all partitioned tables that use list methods. To identify existing tables partitioned with list methods, issue the following SQL statement:

```
SELECT u.name AS OWNER, o.name AS TABLE_NAME
       FROM sys.user$ u, sys.obj$ o, sys.partobj$ po
       WHERE po.parttype = 4
              AND o.obj# = po.obj# AND o.owner# = u.user#;
```

If you do not need to preserve the table data, then, for each table represented by the OWNER.TABLE_NAME columns, simply drop the table:

```
DROP TABLE OWNER.TABLE_NAME;
```

However, if you need to preserve the table data, then copy the data into non-partitioned tables, or copy the data into tables partitioned by range, hash, or another composite method.

Hash Partitioned Index-Organized Tables

This section describes removing incompatibilities relating to hash partitioned index-organized tables.

Drop All Hash Partitioned Index-Organized Tables Before you downgrade to release 8.1.7 or lower, drop 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, for each table represented by the OWNER.TABLE_NAME columns, simply drop the table:

```
DROP TABLE OWNER.TABLE_NAME;
```

However, if you need to preserve the table data, then 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.

```
CREATE range or non-partitioned index-organized table ... AS SELECT * FROM
```

```
OWNER.TABLE_NAME;  
DROP TABLE OWNER.TABLE_NAME;
```

- Export the table using the Oracle9i 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 8.1.7 Import utility.

PDML ITL Invariants

Before you downgrade to release 8.1.7 or lower, remove all PDML ITL invariants. To identify existing PDML ITL invariants, issue the following SQL statement:

```
SELECT COUNT(*) FROM sys.tab$  
WHERE BITAND(property, 536870912) > 0;
```

If this query returns a result greater than 0, then perform the following steps:

1. Change to the `ORACLE_HOME/rdbms/admin` directory.
2. Run `utlpitl.sql`:

```
SQL> @utlpitl.sql
```

Partitioned Index-Organized Tables with LOBs

This section describes removing incompatibilities relating to partitioned index-organized tables with LOBs.

Drop All LOB Columns in Partitioned Index-Organized Tables Before you downgrade to release 8.1.7 or lower, drop all LOB columns in partitioned index-organized tables. To identify existing partitioned index-organized tables with LOB columns, issue the following SQL statement:

```
SELECT t.OWNER, t.TABLE_NAME, l.COLUMN_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, then, for each column represented by the `COLUMN_NAME` column in the table represented by the `OWNER.TABLE_NAME` columns, simply drop the column:

```
ALTER TABLE OWNER.TABLE_NAME DROP COLUMN COLUMN_NAME;
```

However, if you need to preserve the LOB columns, then you can create corresponding non-partitioned index-organized tables:

```
CREATE non-partitioned index-organized table ... AS SELECT * FROM OWNER.TABLE_
NAME;
DROP TABLE OWNER.TABLE_NAME;
```

Drop All Varray Columns in Partitioned Index-Organized Tables Before you downgrade to release 8.1.7 or lower, drop all varray columns in partitioned index-organized tables. To identify existing partitioned index-organized tables with varray columns, issue the following SQL statement:

```
SELECT v.OWNER, v.PARENT_TABLE_NAME, v.PARENT_TABLE_COLUMN
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, then, for each column represented by the PARENT_TABLE_COLUMN column in the table represented by the OWNER.PARENT_TABLE_NAME columns, simply drop the column:

```
ALTER TABLE OWNER.PARENT_TABLE_NAME DROP COLUMN PARENT_TABLE_COLUMN;
```

However, if you need to preserve the varray columns, then you can create corresponding non-partitioned index-organized tables:

```
CREATE non-partitioned index-organized table ... AS SELECT * FROM OWNER.PARENT_
TABLE_NAME;
DROP TABLE OWNER.PARENT_TABLE_NAME;
```

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';
```

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;
```

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, unique, and foreign key constraints. To identify existing view constraints, issue the following SQL statement:

```
SELECT * FROM dba_constraints WHERE view_related = 'DEPEND_ON_VIEW';
```

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 previous release.

See Also: ["Lowering the COMPATIBLE Initialization Parameter"](#)
on page 5-8

If your database fails to open after lowering the value of the `COMPATIBLE` initialization parameter, then some incompatibilities still exist. If so, reset the `COMPATIBLE` initialization parameter to the higher setting. Remove the incompatibilities and attempt to reset database compatibility again. All incompatibilities with the release to which you are downgrading must be removed before you proceed with the downgrade process.

See Also: "[Remove Incompatibilities](#)" on page 7-2 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 Also: "[Remove Incompatibilities](#)" on page 7-2 if you have not yet removed incompatibilities

Complete the following steps to downgrade your release 9.2 database to the previous Oracle release:

1. Log in to the system as the owner of the release 9.2 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. Start up the instance in `MIGRATE` mode:

```
SQL> STARTUP MIGRATE
```

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

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

```
SQL> SPOOL downgrade.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` command:

```
SQL> SET ECHO ON
```

7. Run `dold_release.sql`, where `old_release` refers to the release to which you are downgrading. See [Table 7-1](#) 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 7-1 Downgrade Scripts

Downgrading To	Run Script
9.0.1	d0900010.sql
8.1.7	d0801070.sql

Note: If the release to which you are downgrading is not included in [Table 7-1](#), 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 release 9.2.
- You must run the script in the release 9.2 environment.
- You only need to run one script, even if your downgrade spans more than one release. For example, if you are downgrading to release 8.1.7, then you only need to run `d0801070.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.

8. 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 6; the suggested name was `downgrade.log`. Correct any problems you find in this file and rerun the appropriate downgrade script if necessary.

If you issued a `SET ECHO ON` command, then you may want to issue a `SET ECHO OFF` command now:

```
SQL> SET ECHO OFF
```

9. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

If you are downgrading a cluster database, then shut down all instances.

10. Exit SQL*Plus.

11. If you are downgrading to release 9.0.1, then copy the following files from the release 9.2 Oracle home to the release 9.0.1 Oracle home:

Component	Copy from Release 9.2 Oracle Home	Copy to Previous Oracle Home
JServer JAVA Virtual Machine	<code>ORACLE_HOME/javavm/install/jvmd901.sql</code>	<code>ORACLE_HOME/javavm/install</code>
Oracle XDK for Java	<code>ORACLE_HOME/xdk/admin/xml901.sql</code>	<code>ORACLE_HOME/xdk/admin</code>
Messaging Gateway	<code>ORACLE_HOME/mgw/admin/mgwd901.sql</code>	<code>ORACLE_HOME/mgw/admin</code>
Oracle Workspace Manager	<code>ORACLE_HOME/rdbms/admin/owmd901.plb</code>	<code>ORACLE_HOME/rdbms/admin</code>

If you are downgrading to release 8.1.7, then copy the following files from the release 9.2 Oracle home to the release 8.1.7 Oracle home:

Component	Copy from Release 9.2 Oracle Home	Copy to Previous Oracle Home
JServer JAVA Virtual Machine	<code>ORACLE_HOME/javavm/install/jvmd817.sql</code>	<code>ORACLE_HOME/javavm/install</code>
Oracle XDK for Java	<code>ORACLE_HOME/xdk/admin/xml817.sql</code>	<code>ORACLE_HOME/xdk/admin</code>

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: If you are downgrading a cluster database, then perform this step on all nodes in which this cluster database has instances configured.

See Also: Your operating system-specific Oracle9i 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 OracleServiceSID Oracle service of the release 9.2 database, where SID is the instance name.

For example, if your SID is ORCL, then enter the following at a command prompt:

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

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

- b. Delete the Oracle service at a command prompt by issuing the ORADIM command. For example, if your SID is ORCL, then enter the following command:

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

- c. Create the Oracle service of the database to which you are downgrading at a 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 database being downgraded.

<i>PASSWORD</i>	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.
<i>USERS</i>	is the maximum number of users who can be granted SYSDBA and SYSOPER privileges.
<i>ORACLE_HOME</i>	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 8.1.7, 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
```

14. 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 Oracle9i 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 Oracle9i 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.

15. Copy configuration files from the release 9.2 Oracle home directory to the Oracle home of the release to which you are downgrading:
 - a. Copy your parameter file from the release 9.2 Oracle home to the Oracle home of the release to which you are downgrading. By default Oracle looks for the parameter file in `ORACLE_HOME/dbs` on UNIX platforms and in `ORACLE_HOME\database` on Windows operating systems. The initialization parameter file can reside anywhere you wish, but it should not reside in the release 9.2 Oracle home.
 - b. If your parameter file has an `IFILE` (include file) entry and the file specified in the `IFILE` entry resides within the release 9.2 Oracle home directory, then copy the file specified by the `IFILE` entry to the Oracle home of the release to which you are downgrading. The file specified in the `IFILE` entry contains additional initialization parameters. After you copy this file, edit the parameter file to point to its new location.
 - c. If you have a password file that resides within the release 9.2 Oracle home directory, 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 platforms, the default password file is `ORACLE_HOME/dbs/orapwsid`. On Windows operating systems, the default password file is `ORACLE_HOME\database\pwsid.ora`. On both UNIX platforms and Windows operating systems, `sid` is your Oracle instance ID.

Note: If you are downgrading a cluster database, then perform this step on all nodes in which this cluster database has instances configured.

- If you are downgrading to release 9.0.1, then set the `CLUSTER_DATABASE` initialization parameter to `false`.
- If you are downgrading to release 8.1.7, then set the `PARALLEL_SERVER` initialization parameter to `false`.

After the downgrade, you must set the appropriate initialization parameter back to `true`.

16. Add the following initialization parameters to your parameter file:

```
_SYSTEM_TRIG_ENABLED = false
JOB_QUEUE_PROCESSES = 0
AQ_TM_PROCESSES = 0
```

If you are downgrading to release 9.0.1, then add the following additional initialization parameter to your parameter file:

```
NLS_LENGTH_SEMANTICS = BYTE
```

These initialization parameters should be removed from your parameter file after the downgrade is complete.

17. At a system prompt, change to the `ORACLE_HOME/rdbms/admin` directory of the previous release.
18. Start SQL*Plus.

Note: If you are downgrading to release 8.1.7, then start Server Manager. Do *not* start SQL*Plus.

19. Connect to the database instance as a user with SYSDBA privileges.
20. 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.

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

```
SPOOL old_scripts.log
```

If you want to see the complete detailed output of the scripts you will run, then you can also issue a `SET ECHO ON` command:

```
SET ECHO ON
```

22. 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 1-11 for more information about changing word size

23. Run `catalog.sql`:

```
@catalog.sql
```

24. Run `catproc.sql`:

```
@catproc.sql
```

25. If you are downgrading to release 8.1.7, then run `catrep.sql`:

```
@catrep.sql
```

26. If you are downgrading a cluster database, then complete the following steps:

- a. If you are downgrading to release 9.0.1, then run `catclust.sql`:

```
@catclust.sql
```

- b. If you are downgrading to release 8.1.7, then run `catparr.sql`:

```
@catparr.sql
```

27. You may need to run one or more catalog scripts supplied with the release to which you are downgrading. For example, to re-create Heterogeneous Services data dictionary views, tables, and packages, run `caths.sql`:

```
@caths.sql
```

28. If the database being downgraded has JServer JAVA Virtual Machine installed, then run the appropriate downgrade script (copied to the previous Oracle home in Step 11) to complete the JServer JAVA Virtual Machine downgrade. When you run the script, replace `ORACLE_HOME` with the full path of the previous Oracle home directory.

If you are downgrading to release 9.0.1, then run the following script:

```
@ORACLE_HOME/javavm/install/jvmd901.sql
```

If you are downgrading to release 8.1.7, then run the following script:

```
@ORACLE_HOME/javavm/install/jvmd817.sql
```

29. If the database being downgraded has Oracle XDK for Java installed, then run the appropriate downgrade script (copied to this directory in Step 11) to complete the Oracle XDK for Java downgrade. When you run the script, 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 9.0.1, then run the following script:

```
@ORACLE_HOME/xdk/admin/xml901.sql
```

If you are downgrading to release 8.1.7, then run the following script:

```
@ORACLE_HOME/xdk/admin/xml817.sql
```

30. If the database being downgraded has Messaging Gateway installed, then run the appropriate downgrade script (copied to this directory in Step 11) to complete the Messaging Gateway downgrade. When you run the script, 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 9.0.1, then run the following script:

```
@ORACLE_HOME/mgw/admin/mgwd901.sql
```

31. If the database being downgraded has Oracle Workspace Manager installed, then run the appropriate downgrade script (copied to this directory in Step 11) to complete the Oracle Workspace Manager downgrade. When you run the script, 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 9.0.1, then run the following script:

```
@ORACLE_HOME/rdbms/admin/owmd901.plb
```

32. 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`.

33. 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 21; the suggested name was `catoutd2.log`. Correct any problems you find in this file and rerun the appropriate script if necessary.

If you issued a `SET ECHO ON` command, then you may want to issue a `SET ECHO OFF` command now:

```
SET ECHO OFF
```

34. 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.

- 35.** Exit Server Manager or SQL*Plus, depending on which you started in Step 18.
- 36.** Remove the initialization parameters from your parameter file that you added in Step 16.

Your database is now downgraded.

Database Migration Using Export/Import

This chapter guides you through the process of upgrading and downgrading data in an Oracle database using the Export and Import utilities. This chapter covers the following topics:

- [Export Dump File Compatibility](#)
- [Source Database and Target Database](#)
- [Upgrade the Source Database Using Export/Import](#)

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

Export Dump File Compatibility

Export dump files can be imported into all future releases of Oracle. For example, an Oracle7 export dump file can be imported by the release 8.1.7, release 9.0.1, and release 9.2 Import utilities.

Export dump files, however, are *not* downward compatible with the Import utilities of previous Oracle releases. That is, exported data *cannot* be imported by the Import utilities of previous Oracle releases. For example, a release 8.1.7 export dump file cannot be imported by the release 8.0.6 Import utility, and an Oracle9i export dump file cannot be imported by the Oracle7 Import utility.

The contents of a database can be imported into a previous Oracle release if you use the Export and Import utilities of the previous release to export and import the data. [Table 8–1](#) details this support.

Table 8–1 Backward Compatibility Support for Export/Import

To Export Data From	Import Into	Use Export/Import Utilities For
Release 9.2 Release 9.0.1 Release 8.1.7 Release 8.0.6	Release 7.3.4	Release 7.3.4 Note: Run the <code>catexp7.sql</code> script before exporting.
Release 9.2 Release 9.0.1 Release 8.1.7	Release 8.0.6	Release 8.0.6
Release 9.2 Release 9.0.1	Release 8.1.7	Release 8.1.7
Release 9.2	Release 9.0.1	Release 9.0.1

As [Table 8–1](#) indicates, to export data from a release 8.0 or higher database into an Oracle7 database, you must first run the `catexp7.sql` script on the release 8.0 or higher database before using the Oracle7 Export utility to export the data.

You do *not* need to run the `catexp7.sql` script if you are exporting data from a release 8.1 or higher database into a release 8.0 database.

Export/Import Usage on Data Incompatible with a Previous Release

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.

For example, partitioned tables are not exported by the Oracle7 Export utility. If you need to move a partitioned table from a release 8.0 or higher database into 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 and higher. 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.

Source Database and Target Database

The **source** database is the database containing the data to be exported. The **target** database is the database into which you are importing the exported data.

Export Utility Requirements

To 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, which is a new Oracle9*i* database. The new Oracle9*i* database must be created and operational before the Import utility can import the exported data.

For example, if you are upgrading to release 9.2 from release 7.3.4, then use the Export utility for release 7.3.4.

Import Utility Requirements

To upgrade a database, use the Import utility shipped with the release of the target database, which is a new Oracle9*i* database. For example, if you are upgrading to release 9.2 from release 7.3.4, then use the Import utility for release 9.2.

Upgrade the Source Database Using Export/Import

To upgrade a database using the Export/Import utilities, complete the following steps:

1. Export data from 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.

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 new Oracle9i target database after the import is complete.

2. Install the new Oracle9i software. Installation is operating system-specific. Installation steps for Oracle9i are covered in your operating system-specific Oracle documentation.
3. If the new Oracle9i database will have the same name as the existing source database, then shut down the existing database before creating the new Oracle9i database.
4. Create the new Oracle9i target database.

See Also: *Oracle9i Database Administrator's Guide* for information about creating an Oracle9i database

5. Start SQL*Plus in the new Oracle9i environment.
6. Connect to the database instance as a user with SYSDBA privileges.
7. Start an Oracle9i 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 Oracle9i 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 Import utility of the new Oracle9i database 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 import, 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 Oracle9i 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 new Oracle9i database prior to making it available to users. See Step 1 on page 8-4 for more information.
13. Complete the procedures described in [Chapter 4, "After Upgrading a Database"](#).

Changes to Initialization Parameters and the Data Dictionary

This appendix lists changes to initialization parameters and the data dictionary across different releases of Oracle. This appendix also discusses compatibility issues with certain initialization parameters.

This appendix covers the following topics:

- [Initialization Parameter Changes](#)
- [Compatibility Issues with Initialization Parameters](#)
- [Static Data Dictionary View Changes](#)
- [Dynamic Performance View Changes](#)

Note: This appendix does not list changes to initialization parameters and the data dictionary that occurred in release 8.0. If you are upgrading from Oracle7, then see "[Changes to Initialization Parameters and the Data Dictionary in Release 8.0](#)" on page D-39 in addition to the changes outlined in this appendix.

Initialization Parameter Changes

The following sections list changes to initialization parameters across different releases of Oracle:

- [Deprecated Initialization Parameters](#)
- [Obsolete Initialization Parameters](#)

See Also: The "What's New in Oracle9i Database Reference" section of *Oracle9i Database Reference* for a list of new initialization parameters in Oracle9i

Deprecated Initialization Parameters

The following sections list initialization parameters that have been deprecated. A deprecated parameter behaves the same way as a regular parameter, except that a warning message is displayed at instance startup if a deprecated parameter is specified in a parameter file. In addition, all deprecated parameters are logged to the alert log at instance startup:

- [Initialization Parameters Deprecated in Release 9.2](#)
- [Initialization Parameters Deprecated in Release 9.0.1](#)

Initialization Parameters Deprecated in Release 9.2

The following initialization parameters were deprecated in release 9.2:

Deprecated	In Favor Of
DRS_START	DG_BROKER_START

Initialization Parameters Deprecated in Release 9.0.1

The following initialization parameters were deprecated in release 9.0.1:

Deprecated	In Favor Of
MTS_CIRCUITS	CIRCUITS
MTS_DISPATCHERS	DISPATCHERS
MTS_MAX_DISPATCHERS	MAX_DISPATCHERS
MTS_MAX_SERVERS	MAX_SHARED_SERVERS

Deprecated	In Favor Of
MTS_SERVERS	SHARED_SERVERS
MTS_SESSIONS	SHARED_SERVER_SESSIONS
PARALLEL_SERVER	CLUSTER_DATABASE
PARALLEL_SERVER_INSTANCES	CLUSTER_DATABASE_INSTANCES

Obsolete Initialization Parameters

The following sections list initialization parameters that have been made obsolete:

- [Initialization Parameters Obsolete in Release 9.2](#)
- [Initialization Parameters Obsolete in Release 9.0.1](#)
- [Initialization Parameters Obsolete in Release 8.1](#)

Note: An attempt to start a release 9.2 database using one or more of these obsolete initialization parameters will succeed, but a warning will be returned and recorded in the alert log.

Initialization Parameters Obsolete in Release 9.2

The following initialization parameters were made obsolete in release 9.2:

DISTRIBUTED_TRANSACTIONS	MAX_TRANSACTION_BRANCHES
PARALLEL_BROADCAST_ENABLED	STANDBY_PRESERVES_NAMES

Initialization Parameters Obsolete in Release 9.0.1

The following initialization parameters were made obsolete in release 9.0.1:

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	OPS_INTERCONNECTS

OPTIMIZER_PERCENT_PARALLEL SORT_MULTIBLOCK_READ_COUNT
TEXT_ENABLE

Initialization Parameters Obsolete in Release 8.1

The following initialization parameters were made obsolete in release 8.1:

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_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_LOCKS
LM_PROCS	LM_RESS
LOCK_SGA_AREAS	LOG_ARCHIVE_BUFFER_SIZE
LOG_ARCHIVE_BUFFERS	LOG_BLOCK_CHECKSUM
LOG_FILES	LOG_SIMULTANEOUS_COPIES
LOG_SMALL_ENTRY_MAX_SIZE	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	

Compatibility Issues with Initialization Parameters

The lists of deprecated and obsolete initialization parameters earlier in this appendix show changes to 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 Oracle9i, 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` initialization parameter.

See Also: *Oracle9i Database Reference* and *Oracle9i Database Performance Tuning 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 and higher, 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` 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 Also: `LOG_CHECKPOINT_TIMEOUT` in *Oracle9i Database Reference*

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 5-30 for more information.

The `DML_LOCKS` Parameter

Oracle9i 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 Oracle9i 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 Oracle9i fewer transactions are used per rollback segment than are used in Oracle7. Consequently, `DML_LOCKS` has a lower default value in Oracle9i. 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 not set, 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. If you are upgrading from release 8.0.6 or earlier, then 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 upgrade.

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 upgrade.

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 previous releases, this allocation was from the shared pool. If you are upgrading from release 8.0.6 or earlier, 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 Tuning 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 upgrade to release 8.1 or higher.

Retaining Parameter Settings without Modifications

You do not alter the parameters from their previous settings:

Table A-1 *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 A-2 *Using `PARALLEL_AUTOMATIC_TUNING`*

Parameter	Setting
<code>PARALLEL_AUTOMATIC_TUNING</code>	Set to <code>true</code> .
<code>SHARED_POOL_SIZE</code>	Set to a small value that accounts for all clients except parallel execution.
<code>LARGE_POOL_SIZE</code>	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 A-3 *Using `PARALLEL_AUTOMATIC_TUNING` and Setting `LARGE_POOL_SIZE`*

Parameter	Setting
<code>PARALLEL_AUTOMATIC_TUNING</code>	Set to <code>true</code> .
<code>SHARED_POOL_SIZE</code>	Set to a small value that accounts for all clients except parallel execution.
<code>LARGE_POOL_SIZE</code>	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 A-4 Using PARALLEL_AUTOMATIC_TUNING without Modifying SHARED_POOL_SIZE

Parameter	Setting
PARALLEL_AUTOMATIC_TUNING	Set to <code>true</code> .
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 upgrade, 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 Oracle9i, 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)
- `standby1` (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=standby1 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.

Static Data Dictionary View Changes

The following sections list changes to static data dictionary views across different releases of Oracle:

- [Deprecated Static Data Dictionary Views](#)
- [Obsolete Static Data Dictionary Views](#)
- [Static Data Dictionary Views with Renamed Columns](#)
- [Static Data Dictionary Views with Dropped Columns](#)
- [Static Data Dictionary Views with Columns That May Return Nulls](#)

See Also: The "What's New in Oracle9i Database Reference" section of *Oracle9i Database Reference* for a list of new static data dictionary views in Oracle9i

Deprecated Static Data Dictionary Views

The following sections list static data dictionary views that have been deprecated:

- [Static Data Dictionary Views Deprecated in Release 9.2](#)
- [Static Data Dictionary Views Deprecated in Release 9.0.1](#)
- [Static Data Dictionary Views Deprecated in Release 8.1](#)

Static Data Dictionary Views Deprecated in Release 9.2

The following static data dictionary views were deprecated in release 9.2:

Deprecated	In Favor Of
ALL_RULESETS	ALL_RULE_SETS
DBA_RULESETS	DBA_RULE_SETS
USER_RULESETS	USER_RULE_SETS

Static Data Dictionary Views Deprecated in Release 9.0.1

The following static data dictionary views were deprecated in release 9.0.1:

Deprecated	In Favor Of
ALL_REGISTERED_SNAPSHOTS	ALL_REGISTERED_MVIEWS
ALL_SNAPSHOT_LOGS	ALL_BASE_TABLE_MVIEWS ALL_MVIEW_LOGS
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_LOGS	DBA_BASE_TABLE_MVIEWS DBA_MVIEW_LOGS
DBA_SNAPSHOT_REFRESH_TIMES	DBA_MVIEW_REFRESH_TIMES
USER_REGISTERED_SNAPSHOTS	USER_REGISTERED_MVIEWS
USER_SNAPSHOT_LOGS	USER_BASE_TABLE_MVIEWS USER_MVIEW_LOGS
USER_SNAPSHOT_REFRESH_TIMES	USER_MVIEW_REFRESH_TIMES

Static Data Dictionary Views Deprecated in Release 8.1

The following static data dictionary views were deprecated in release 8.1:

Deprecated	In Favor Of
ALL_SNAPSHOTS	ALL_MVIEWS
ALL_SUMMARIES	ALL_MVIEW_ANALYSIS

Deprecated	In Favor Of
ALL_SUMMARY_AGGREGATES	ALL_MVIEW_AGGREGATES
ALL_SUMMARY_DETAIL_TABLES	ALL_MVIEW_DETAIL_RELATIONS
ALL_SUMMARY_JOINS	ALL_MVIEW_JOINS
ALL_SUMMARY_KEYS	ALL_MVIEW_KEYS
DBA_SNAPSHOTS	DBA_MVIEWS
DBA_SUMMARIES	DBA_MVIEW_ANALYSIS
DBA_SUMMARY_AGGREGATES	DBA_MVIEW_AGGREGATES
DBA_SUMMARY_DETAIL_TABLES	DBA_MVIEW_DETAIL_RELATIONS
DBA_SUMMARY_JOINS	DBA_MVIEW_JOINS
DBA_SUMMARY_KEYS	DBA_MVIEW_KEYS
USER_SNAPSHOTS	USER_MVIEWS
USER_SUMMARIES	USER_MVIEW_ANALYSIS
USER_SUMMARY_AGGREGATES	USER_MVIEW_AGGREGATES
USER_SUMMARY_DETAIL_TABLES	USER_MVIEW_DETAIL_RELATIONS
USER_SUMMARY_JOINS	USER_MVIEW_JOINS
USER_SUMMARY_KEYS	USER_MVIEW_KEYS

Obsolete Static Data Dictionary Views

The following sections list static data dictionary views that have been made obsolete:

- [Static Data Dictionary Views Obsolete in Release 8.1](#)

Static Data Dictionary Views Obsolete in Release 8.1

The following static data dictionary views were made obsolete in release 8.1:

ALL_LABELS

Static Data Dictionary Views with Renamed Columns

The following sections list static data dictionary views with renamed columns:

- [Static Data Dictionary Views with Renamed Columns in Release 9.0.1](#)

Static Data Dictionary Views with Renamed Columns in Release 9.0.1

The static data dictionary view columns listed in [Table A-5](#) were renamed in release 9.0.1:

Table A-5 *Static Data Dictionary Views with Renamed Columns in Release 9.0.1*

Static Data Dictionary View	Pre-Release 9.0.1 Column Name	Release 9.0.1 and Higher Column Name
DBA_RSRC_PLAN_DIRECTIVES	MAX_ACTIVE_SESS_TARGET_P1	ACTIVE_SESS_POOL_P1
DBA_RSRC_PLANS	MAX_ACTIVE_SESS_TARGET_MTH	ACTIVE_SESS_POOL_MTH

Static Data Dictionary Views with Dropped Columns

The following sections list static data dictionary views with dropped columns:

- [Static Data Dictionary Views with Dropped Columns in Release 9.0.1](#)
- [Static Data Dictionary Views with Dropped Columns in Release 8.1](#)

Static Data Dictionary Views with Dropped Columns in Release 9.0.1

The following static data dictionary view columns were dropped in release 9.0.1:

Static Data Dictionary View	Dropped Columns
DBA_RSRC_PLAN_DIRECTIVES	MAX_ACTIVE_SESS_TARGET_P1
DBA_RSRC_PLANS	MAX_ACTIVE_SESS_TARGET_MTH

Static Data Dictionary Views with Dropped Columns in Release 8.1

The following static data dictionary view columns were dropped in release 8.1:

Static Data Dictionary Views	Dropped Columns
DBA_AUDIT_OBJECT	OBJECT_LABEL
USER_AUDIT_OBJECT	SESSION_LABEL
DBA_AUDIT_SESSION	SESSION_LABEL
USER_AUDIT_SESSION	

Static Data Dictionary Views	Dropped Columns
DBA_AUDIT_STATEMENT	SESSION_LABEL
USER_AUDIT_STATEMENT	
DBA_AUDIT_TRAIL	OBJECT_LABEL
USER_AUDIT_TRAIL	SESSION_LABEL
DBA_CONTEXT	ATTRIBUTE
ALL_IND_COLUMNS	COLUMN_EXPRESSION
DBA_IND_COLUMNS	
USER_IND_COLUMNS	
ALL_JOBS	CLEARANCE_HI
DBA_JOBS	CLEARANCE_LO
USER_JOBS	CURRENT_SESSION_LABEL
ALL_REFS	HAS_REFERENTIAL_CONS
DBA_REFS	REFERENTIAL_CONS_NAME
USER_REFS	

Static Data Dictionary Views with Columns That May Return Nulls

Starting with release 8.1, the static data dictionary view columns listed in [Table A-6](#) may return nulls; in previous releases, these columns could not return nulls. If an application requires non-null values for one or more of these columns, then modify the application accordingly:

Table A-6 Static Data Dictionary Views with 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.
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.

Table A–6 (Cont.) Static Data Dictionary Views with Columns That May Return Nulls in Release 8.1

Static Data Dictionary Views	Columns	Explanation
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.
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.

Dynamic Performance View Changes

The following sections list changes to dynamic performance views (V\$ views) across different releases of Oracle:

- [Deprecated Dynamic Performance Views](#)
- [Obsolete Dynamic Performance Views](#)
- [Dynamic Performance Views with Renamed Columns](#)
- [Dynamic Performance Views with Dropped Columns](#)

See Also: The "What's New in Oracle9i Database Reference" section of *Oracle9i Database Reference* for a list of new dynamic performance views in Oracle9i

Deprecated Dynamic Performance Views

The following sections list dynamic performance views that have been deprecated:

- [Dynamic Performance Views Deprecated in Release 9.2](#)

- [Dynamic Performance Views Deprecated in Release 9.0.1](#)

Dynamic Performance Views Deprecated in Release 9.2

The following dynamic performance views were deprecated in release 9.2:

Deprecated	In Favor Of
GV\$SORT_USAGE	GV\$TEMPSEG_USAGE
V\$SORT_USAGE	V\$TEMPSEG_USAGE

Dynamic Performance Views Deprecated in Release 9.0.1

The following dynamic performance views were deprecated in release 9.0.1:

Deprecated	In Favor Of
GV\$BSP	GV\$CR_BLOCK_SERVER
GV\$CLASS_PING	GV\$CLASS_CACHE_TRANSFER
GV\$DLM_ALL_LOCKS	GV\$GES_ENQUEUE
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\$MAX_ACTIVE_SESS_TARGET_MTH	GV\$ACTIVE_SESS_POOL_MTH
GV\$MTS	GV\$SHARED_SERVER_MONITOR
GV\$PING	GV\$CACHE_TRANSFER
GV\$TEMP_PING	GV\$TEMP_CACHE_TRANSFER
V\$BSP	V\$CR_BLOCK_SERVER

Deprecated	In Favor Of
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\$MAX_ACTIVE_SESS_TARGET_MTH	V\$ACTIVE_SESS_POOL_MTH
V\$MTS	V\$SHARED_SERVER_MONITOR
V\$PING	V\$CACHE_TRANSFER
V\$TEMP_PING	V\$TEMP_CACHE_TRANSFER

Obsolete Dynamic Performance Views

The following sections list dynamic performance views that have been made obsolete:

- [Dynamic Performance Views Obsolete in Release 9.2](#)
- [Dynamic Performance Views Obsolete in Release 9.0.1](#)
- [Dynamic Performance Views Obsolete in Release 8.1](#)

Dynamic Performance Views Obsolete in Release 9.2

The following dynamic performance views were made obsolete in release 9.2:

GV\$LOADCSTAT	V\$LOADCSTAT
GV\$LOADTSTAT	V\$LOADTSTAT

Dynamic Performance Views Obsolete in Release 9.0.1

The following dynamic performance views were made obsolete in release 9.0.1:

GV\$TARGETRBA

V\$TARGETRBA

Dynamic Performance Views Obsolete in Release 8.1

The following dynamic performance views were made obsolete in release 8.1:

GV\$CURRENT_BUCKET

V\$CURRENT_BUCKET

GV\$RECENT_BUCKET

V\$RECENT_BUCKET

Dynamic Performance Views with Renamed Columns

The following sections list dynamic performance views with renamed columns:

- [Dynamic Performance Views with Renamed Columns in Release 9.2](#)
- [Dynamic Performance Views with Renamed Columns in Release 9.0.1](#)
- [Dynamic Performance Views with Renamed Columns in Release 8.1](#)

Dynamic Performance Views with Renamed Columns in Release 9.2

The dynamic performance view columns listed in [Table A-7](#) were renamed in release 9.2:

Table A-7 *Dynamic Performance Views with Renamed Columns in Release 9.2*

Dynamic Performance View	Pre-Release 9.2 Column Name	Release 9.2 and Higher Column Name
GV\$ARCHIVE_DEST and V\$ARCHIVE_DEST	MANIFEST	REGISTER
	REGISTER	REMOTE_TEMPLATE
GV\$DATABASE and V\$DATABASE	STANDBY_MODE	PROTECTION_MODE
GV\$LOGMNR_CALLBACK and V\$LOGMNR_CALLBACK	CALLBACK_STATE	STATE
	CALLBACK_TYPE	TYPE
	CALLBACK_CAPABILITY	CAPABILITY
GV\$LOGMNR_REGION and V\$LOGMNR_REGION	ID	MEMSTATE
	CURRENT_STATE	STATE

Dynamic Performance Views with Renamed Columns in Release 9.0.1

The dynamic performance view columns listed in [Table A-8](#) were renamed in release 9.0.1:

Table A-8 *Dynamic Performance Views with Renamed Columns in Release 9.0.1*

Dynamic Performance View	Pre-Release 9.0.1 Column Name	Release 9.0.1 and Higher Column Name
GV\$RSRC_CONSUMER_GROUP and V\$RSRC_CONSUMER_GROUP	SESSIONS_QUEUED	QUEUE_LENGTH

Dynamic Performance Views with Renamed Columns in Release 8.1

The dynamic performance view columns listed in [Table A-9](#) were renamed in release 8.1:

Table A-9 *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 with Dropped Columns

The following sections list dynamic performance views with dropped columns. If an application requires one or more of these columns, then modify the application accordingly:

- [Dynamic Performance Views with Dropped Columns in Release 9.2](#)
- [Dynamic Performance Views with Dropped Columns in Release 9.0.1](#)
- [Dynamic Performance Views with Dropped Columns in Release 8.1](#)

Dynamic Performance Views with Dropped Columns in Release 9.2

The following dynamic performance view columns were dropped in release 9.2:

Dynamic Performance View	Dropped Columns
GV\$DATABASE and V\$DATABASE	STANDBY_MODE
GV\$LOGMNR_CALLBACK and V\$LOGMNR_CALLBACK	FUNC_NAME CALLBACK_ID CALLBACK_RESULT_SIZE CALLBACK_STATE CALLBACK_TYPE CALLBACK_CAPABILITY NUMBER_INVOKED
GV\$LOGMNR_REGION and V\$LOGMNR_REGION	ID CURRENT_STATE

Dynamic Performance Views with Dropped Columns in Release 9.0.1

The following dynamic performance view columns were dropped in release 9.0.1:

Dynamic Performance View	Dropped Columns
GV\$LOGMNR_CONTENTS and V\$LOGMNR_CONTENTS	PH1_NAME PH1_REDO PH1_UNDO PH2_NAME PH2_REDO PH2_UNDO PH3_NAME PH3_REDO PH3_UNDO PH4_NAME PH4_REDO PH4_UNDO PH5_NAME PH5_REDO PH5_UNDO
GV\$RSRC_CONSUMER_GROUP and V\$RSRC_CONSUMER_GROUP	SESSIONS_QUEUED

Dynamic Performance Views with Dropped Columns in Release 8.1

The following dynamic performance view columns were dropped 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

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

Upgrade Considerations for Oracle Net Services

This appendix describes coexistence and upgrade issues for Oracle Net Services. This appendix covers the following topics:

- [Overview of Unsupported Oracle Net Services Features](#)
- [Unsupported Parameters and Control Utility Commands](#)
- [Client and Database Coexistence Issues](#)
- [Using the Oracle Net Manager to Handle Compatibility Issues](#)
- [Upgrading to Oracle Net Services](#)
- [Using Oracle Names Version 9](#)

Overview of Unsupported Oracle Net Services Features

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](#)
- [Protocols](#)

Identix and SecurID Authentication Methods

If you are using Identix or SecurID authentication, provided by Oracle Advanced Security, Oracle Corporation recommends upgrading 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: *Oracle9i 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.

Prespawnd Dedicated Servers

Prespawnd dedicated server processes are no longer supported. Instead, configure shared server (formerly named multi-threaded server) to improve scalability and system resource usage.

Protocols

Protocol addresses using the SPX or LU6.2 protocol must be replaced. Oracle Net provides support for the following network protocols:

- TCP/IP
- TCP/IP with SSL
- Named Pipes

See Also: *Oracle9i Net Services Reference Guide* for protocol parameter information

Unsupported Parameters and Control Utility Commands

See Also: *Oracle9i Net Services Reference Guide* for further information about unsupported configuration parameters and control utility commands

Client and Database Coexistence Issues

Clients and database servers require compatible releases of Oracle Net Services or Net8. For example, an Oracle9i client requires an installation of Oracle Net Services, and an Oracle9i 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 upgrading is appropriate for your environment:

- [Oracle9i Database Connections](#)
- [Oracle8 or Oracle7 Database Connections](#)
- [Oracle Names](#)

Oracle9i Database Connections

Connect descriptors, created for connections to an Oracle9i or an Oracle8i database, identify a database by its service name with the `SERVICE_NAME` parameter.

A connect descriptor to an Oracle9i or Oracle8i database uses the parameter `SERVICE_NAME`, as shown in the following example:

```
sales=
(DESCRIPTION=
 (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(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 B-7.

See Also: *Oracle9i Net Services Administrator's Guide* for information about database identification by `SERVICE_NAME` rather than `SID`

Consider the following questions for an environment with release 8.0 clients connecting to an Oracle9i 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* Oracle9i 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 Oracle9i 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-server)(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 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: *Oracle9i Net Services Administrator's Guide* for information about database identification by SID

Consider the following questions for an environment with Oracle9i clients connecting to a 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* 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 release 8.0 database?*

Yes. The client requires an installation of Oracle Net in its Oracle home and the release 8.0 database requires an installation of Net8 Server in its Oracle home.

Oracle Names

If you upgrade all or part of your network to Oracle9i, you should upgrade all the Oracle Names Servers in the region to version 9.

- *Can my release 8.0 clients use Oracle Names version 9 to resolve service names?*

Yes.

- *Can my release 8.0 clients 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 upgrading an existing Oracle Names configuration to directory naming, as described in the *Oracle9i Net Services Administrator's Guide*.

Using the Oracle Net Manager to Handle Compatibility Issues

Because some parameters are enabled only for release 9i 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 particular release of the database. These options are described in [Table B-1](#).

Table B-1 *Compatibility Options Available with Oracle Net Manager*

Oracle Net Manager Option	Description
Use Options Compatible with Net8 8.0 Clients	<p>Enables you to configure multiple addresses parameters for a client.</p> <p>If selected, enables the <code>SOURCE_ROUTE</code> parameter for pre-release 8.1 clients requiring Oracle Connection Manager connections.</p> <p>If turned off, enables you to use the <code>SOURCE_ROUTE</code>, <code>LOAD_BALANCE</code>, and <code>FAILOVER</code> parameters for release 9i and release 8.1 clients.</p> <p>See Also: <i>Oracle9i Net Services Administrator's Guide</i> for information about configuring address list parameters</p>
Use Oracle8 Release 8.0 Compatible Identification	<p>Enables you to configure parameters specific to a database release in the <code>CONNECT_DATA</code> section of a connect descriptor.</p> <p>If turned on, allows you to enter the SID of the release 8.0 or Oracle7 database.</p> <p>If turned off, enables you to enter the Oracle9i or Oracle8i database service name (<code>SERVICE_NAME</code>).</p> <p>Note: The <i>Advanced Service Options</i> 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 Oracle9i or Oracle8i database service.</p> <p>See Also: <i>Oracle9i Net Services Administrator's Guide</i> for information about configuring advanced connect data parameters</p>

Upgrading to Oracle Net Services

To upgrade from SQL*Net release 2.x to Oracle Net Services 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 on the Database Server](#)

[Step 3: Perform Software Upgrade on the Client](#)

[Step 4: Perform Functional Upgrade](#)

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 B-2 Initialization Parameters for Oracle Net Services

Parameter	Description
<code>SERVICE_NAMES</code>	<p>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:</p> <pre>SERVICE_NAMES = sales.us.acme.com, widgetsales.us.acme.com</pre> <p>If you do not qualify the names in this parameter with a domain, Oracle qualifies them with the value of the <code>DB_DOMAIN</code> parameter. If <code>DB_DOMAIN</code> is not specified, Oracle uses the domain of your local database as it currently exists in the data dictionary.</p> <p>Note: You can change the value of <code>SERVICE_NAMES</code> parameter dynamically with the <code>SQL ALTER SYSTEM</code> when the database is running. See the <i>Oracle9i Database Reference</i> for further information about this parameter</p>
<code>INSTANCE_NAME</code>	<p>Specifies the unique name of this instance. Set the instance name to the value of the Oracle System Identifier (SID).</p>

Step 2: Perform Software Upgrade on the Database Server

To perform a software upgrade 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 a database with the Database Upgrade Assistant if the Oracle Universal Installer detects a pre-release 9.2 database on your system. If you do not want to upgrade 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 9.2 listener

Step 3: Perform Software Upgrade on the Client

To perform a software upgrade on the client, install the latest release of Oracle Net Services from the Oracle Universal Installer to receive the latest executables.

Step 4: Perform Functional Upgrade

After the software is upgraded, it is *not* required to upgrade the configuration files unless you want to use the Oracle9i features. To take advantage of new features, review the following configuration files:

- `sqlnet.ora`
- `tnsnames.ora`
- `listener.ora`
- `protocol.ora`

Replace obsolete or renamed parameters.

See Also: *Oracle9i Net Services Reference Guide* for further information about unsupported configuration parameters

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-server)(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=sales1-server)(PORT=1521))
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521))
  )
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.acme.com)))
```

See Also:

- ["Using the Oracle Net Manager to Handle Compatibility Issues"](#) on page B-7 for information about configuring the service name and multiple address features
- *Oracle9i Net Services Administrator's Guide* for information about multiple addresses

listener.ora

Because instance information is registered with the listener in release 9i, 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 B-3 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 <code>INSTANCE_NAME</code> parameter in the initialization parameter file.
GLOBAL_DBNAME	The global database name is comprised of the database name and database domain name. You can obtain the <code>GLOBAL_DBNAME</code> value from the <code>SERVICE_NAMES</code> parameter, or from the <code>DB_NAME</code> and <code>DB_DOMAIN</code> 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 Oracle9i Real Application Clusters environment, Oracle Corporation recommends not setting the `GLOBAL_DBNAME` parameter.

See Also: *Oracle9i 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 upgrading an existing Oracle Names configuration to directory naming, as described in *Oracle9i 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 upgrade all of your existing Oracle Names Servers in a region to version 9 by installing Oracle Names version 9 on every existing Oracle Names server node.

Upgrade issues to keep in mind are described in the following sections:

- [Upgrading from Oracle Names Version 2 Using a Database](#)
- [Upgrading from Oracle Names Version 2 with the Dynamic Discovery Option](#)
- [Upgrading from ROSFILES](#)
- [Upgrading Region Checkpoint Files to Domain and Topology Checkpoint Files](#)
- [Reviewing Upgrade Checklist](#)

Upgrading from Oracle Names Version 2 Using a Database

To upgrade 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 upgrade 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;
```


Upgrading from Oracle Names Version 2 with the Dynamic Discovery Option

The procedure to upgrade 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 Upgrade](#)
- [Region Database Upgrade](#)

Non-Region Database Upgrade

If you upgrade 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 *Oracle9i Net Services Administrator's Guide*.

Region Database Upgrade

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: *Oracle9i 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...
<ol style="list-style-type: none">1. Start the Oracle Net Manager:<ul style="list-style-type: none">-On UNIX, run <code>netmgr</code> at <code>\$ORACLE_HOME/bin</code>.-On Windows platforms, choose Start > Programs > Oracle - HOME_NAME > Configuration and Migration Tools > Net Manager2. 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 <code>tnsnames.ora</code> file in the File field created in Step 1.8. Choose Execute.9. Choose File > Save Network Configuration.	<p>From the command line, enter:</p> <pre>namesctl NAMECTL> LOAD_TNSNAMES file_name</pre>

Upgrading from ROSFILES

Oracle Names version 8 and higher do not support older configurations that use Resource Object Store (ROS) files (ROSFILES). ROSFILES must be upgraded 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 upgrade 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\dba\rosbild.sql;
SQL> @oracle_home\dba\nmcbild.sql;
SQL> @oracle_home\dba\rosgrnt.sql;
SQL> @oracle_home\dba\nmcgrnt.sql;
```

Script	Description
<code>rosbild.sql</code>	Builds tables for use by the ROS
<code>nmcbild.sql</code>	Builds tables for use by the Oracle Network Manager Objects (NMO) components
<code>rosgrnt.sql</code>	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.
<code>nmcgrnt.sql</code>	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 **File > Save As**.
 - 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 ["Upgrading from Oracle Names Version 2 Using a Database"](#) on page B-13.

ROSFILES to a tnsnames.ora File

To upgrade 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...
<ol style="list-style-type: none"> 1. Start the Oracle Net Manager. <ul style="list-style-type: none"> -On UNIX, run <code>netmgr</code> at <code>\$ORACLE_HOME/bin</code>. -On Windows platforms, choose Start > Programs > Oracle - HOME_NAME > Configuration and Migration Tools > 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 <code>tnsnames.ora</code> file in the File field. 8. Choose Execute. 9. Choose File > Save Network Configuration. 	<p>From the command line, enter:</p> <pre>namesctl NAMECTL> LOAD_TNSNAMES file_name</pre>

See Also:

- *Oracle Network Manager Administrator's Guide*, release 3.1
- *Oracle Names Administrator's Guide*, version 2

Upgrading 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*i*, 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 9.2.
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 operating systems.

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 9.2.
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: *Oracle9i Net Services Administrator's Guide* for an example `cktop.ora` file

Reviewing Upgrade Checklist

The following checklist is provided to ensure a proper upgrade to Oracle Names version 9.

- ❑ Upgrade 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 *Oracle9i 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.

Migrating from Server Manager to SQL*Plus

This appendix guides you through the process of modifying your Server Manager line mode scripts to work with SQL*Plus. Server Manager is no longer supported in Oracle9i. 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 appendix covers the following topics:

- [Startup Differences](#)
- [Commands](#)
- [Syntax Differences](#)

See Also: *SQL*Plus User's Guide and Reference* for detailed information about SQL*Plus

Note: For brevity, Server Manager line mode is referred to as Server Manager in the rest of this appendix.

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 C-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, then 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 C-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.

Table C-1 (Cont.) Commands Introduced in SQL*Plus Release 8.1

Command	Description
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 C-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 C-2 Server Manager Commands Corresponding to Existing SQL*Plus Commands

Command	Description
CONNECT	Connects to a database using the specified user name.
DESCRIBE	Describes a function, package, package body, procedure, table, view, or object type. 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.
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 C-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 C-3 SQL*Plus Equivalents for Server Manager Commands

Server Manager Commands	SQL*Plus Commands	Description
SET CHARWIDTH SET DATEWIDTH SET LONGWIDTH	COLUMN FORMAT	<p>You can use the <code>COLUMN FORMAT</code> command in SQL*Plus to set the column width of character columns, date columns, and number columns. In your SQL scripts, replace the <code>SET CHARWIDTH</code>, <code>SET DATEWIDTH</code>, and <code>SET LONGWIDTH</code> Server Manager commands with the SQL*Plus <code>COLUMN FORMAT</code> command.</p> <p>Use <code>COLUMN FORMAT</code> for all character columns to be changed. There is no equivalent command to change all character columns with one command.</p> <p>For example, suppose you have the following entry in a SQL script:</p> <pre>SET CHARWIDTH 5</pre> <p>This command sets the width for all character columns to 5 in Server Manager.</p> <p>To specify that a particular column, such as <code>first_name</code>, display with a width of 5 characters, enter the following SQL*Plus command:</p> <pre>COLUMN first_name FORMAT A5</pre> <p>Use <code>COLUMN FORMAT</code> for all character columns to be changed. There is no equivalent command to change all character columns with one command.</p> <p>Use <code>COLUMN FORMAT</code> for all date columns to be changed. There is no equivalent command to change all date columns with one command.</p> <p>Use <code>SET LONG</code> to specify how much of the <code>LONG</code> column to fetch and display.</p>
SET STOPONERROR	WHENEVER SQLERROR WHENEVER OSERROR	<p>Use the <code>WHENEVER SQLERROR</code> and <code>WHENEVER OSERROR</code> 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 <code>SET STOPONERROR</code> command.</p> <p>For both <code>WHENEVER SQLERROR</code> and <code>WHENEVER OSERROR</code>, the <code>EXIT</code> clause directs SQL*Plus to exit, while the <code>CONTINUE</code> clause directs SQL*Plus to continue. Other terms and clauses are also available for these commands.</p>

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 C-11 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_id LIKE '%CLERK'; /* Includes only clerks. */
```

In SQL*Plus, replace this code with either of the following entries:

```
SELECT * FROM hr.employees
      WHERE job_id LIKE '%CLERK';
/* Includes only clerks. */
```

```
SELECT * FROM hr.employees
      WHERE job_id LIKE '%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_id LIKE '%CLERK'; -- Includes only clerks.
```


In SQL*Plus, replace this code with either of the following entries:

```
SELECT * FROM hr.employees
      WHERE job_id LIKE '%CLERK';
-- Includes only clerks.

SELECT * FROM hr.employees
      WHERE job_id LIKE '%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_id LIKE '%CLERK';
```

In SQL*Plus, replace this code with the following entry:

```
-- This statement returns only clerks.
SELECT * FROM hr.employees
      WHERE job_id LIKE '%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 employee_id, first_name, last_name, salary, commission_pct
      FROM hr.employees
      WHERE job_id LIKE '%MAN';
```

Either set `SQLBLANKLINES` to `ON`, or delete the blank lines:

```
SELECT employee_id, first_name, last_name, salary, commission_pct
      FROM hr.employees
      WHERE job_id LIKE '%MAN';
```

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 employee_id, first_name, last_name FROM hr.employees -
WHERE job_id LIKE '%MAN';
```

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 employee_id, first_name, last_name, job_id  
       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
/
```

Upgrading an Oracle7 Database Using the MIG Utility

This appendix describes how to use the MIG utility to manually upgrade an Oracle7 database to the new Oracle9i release. This appendix covers the following topics:

- [Overview of the MIG Utility](#)
- [System Considerations and Requirements for Using the MIG Utility](#)
- [Prepare the Oracle7 Database to be Upgraded](#)
- [Review MIG Utility Command-Line Options](#)
- [Run the MIG Utility](#)
- [MIG Utility Messages](#)
- [Troubleshooting MIG Utility Errors](#)
- [Abandoning the Oracle7 Upgrade](#)
- [Migration Issues for Physical Rowids](#)
- [Changes to Initialization Parameters and the Data Dictionary in Release 8.0](#)

Overview of the MIG Utility

The MIG utility converts the data dictionary and structures of an Oracle7 database into Oracle9i format. To upgrade the database, you first install the Oracle9i software and run the MIG utility on the Oracle7 database. Then, you execute a series of `ALTER DATABASE` statements on the new Oracle9i database and run the `u0703040.sql` upgrade script.

The completion of these procedures results in the conversion of the following Oracle7 structures into structures that can be used by Oracle9i:

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

Outline of the Upgrade Process Using the MIG Utility

The following sections provide an outline of the upgrade process using the MIG utility:

In the Oracle7 Environment

- You run the Oracle9i MIG 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 Oracle9i MIG utility multiple times (without opening the database in Oracle9i) and still be able to return to the Oracle7 database. However, running the MIG utility automatically eliminates the Oracle7 database catalog views (see ["Abandoning the Oracle7 Upgrade"](#) on page D-31).

In the Oracle9i Environment

- You execute an `ALTER DATABASE CONVERT` statement, which creates a new control file based on the convert file generated by the MIG utility, converts all online datafile headers to Oracle9i format, and mounts the Oracle9i database.

The file headers of offline datafiles and read-only tablespaces are not updated during the upgrade. 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 the upgrade; however, they never have to be made read-write.

- You execute an `ALTER DATABASE OPEN RESETLOGS MIGRATE` statement, which automatically converts all objects and users defined in the new dictionary to Oracle9i specifications, and converts all rollback segments to Oracle9i format.

If a database rollback segment is in a tablespace that is offline when the Oracle9i database is opened, then the rollback segment is not converted immediately to Oracle9i database format. Instead, the rollback segment is converted the first time the tablespace is brought online in Oracle9i.

- You run the `u0703040.sql` upgrade 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.

System Considerations and Requirements for Using the MIG Utility

The following sections discuss additional system considerations and requirements for using the MIG utility. These requirements supplement the general upgrade requirements discussed in "[System Considerations and Requirements](#)" on page 3-9.

Space Requirements

Oracle9i executables may require as much as three times the disk space required by Oracle7 executables. This requirement may cause you to run out of disk space during the upgrade. If you are installing Oracle9i 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 Oracle9i and Oracle7 to determine the total system requirements.

The MIG utility requires relatively little temporary space. It needs only enough extra room in the `SYSTEM` tablespace to hold the new Oracle9i 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 Oracle9i data dictionary requires double the

space that its Oracle7 data dictionary required. If necessary, add space to the `SYSTEM` tablespace.

In addition, running scripts such as the `u0703040.sql` upgrade script 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 script. The exact amount of space required to run the 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: Your operating system-specific installation documentation for detailed information about system requirements

Block Size Considerations

The value of the `DB_BLOCK_SIZE` initialization parameter in both the Oracle7 database and in the upgraded Oracle9i database *must* be the same. Oracle9i 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 Oracle9i block size setting meets the following criteria:

- Matches the Oracle7 setting.
- Is at least 2048 bytes (2 KB). The MIG 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 upgrade your database to the new Oracle9i release, not the least of which is deciding whether you will migrate to Oracle Net Services.

See Also: [Appendix B, "Upgrade Considerations for Oracle Net Services"](#) for information about these issues and for instructions on migrating from SQL*Net to Oracle Net Services

Considerations for Replication Environments

You can upgrade an Oracle7 replication environment to Oracle9i. Oracle7 sites can coexist and run successfully with Oracle8, Oracle8i, and Oracle9i sites within the replication environment. However, take special care to accommodate the various replication features implemented on each system.

See Also: [Appendix E, "Database Migration and Compatibility for Replication Environments"](#) for detailed instructions about upgrading 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.

Distributed Database Considerations

When upgrading from Oracle7 in a distributed database configuration, make sure that no pending transactions are in the `DBA_2PC_PENDING` data dictionary view before upgrading the database. Otherwise, when you open the database after the upgrade 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 Database to be Upgraded

Additional preparatory steps are required before you upgrade your Oracle7 database to the new Oracle9i release. Complete the following steps:

1. Log in to the system as the owner of the Oracle home directory of the Oracle7 database being upgraded.
2. Start Server Manager.
3. Connect to the database instance as a user with `SYSDBA` privileges.
4. 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.

5. 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 upgrade the database. You can use the V\$DATAFILE dynamic performance view to find the datafile name based on the datafile number. The MIG 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 the upgrade. Otherwise, these tablespaces will not be available under Oracle9i after the upgrade. Typically, tablespaces that are taken offline by using an ALTER TABLESPACE OFFLINE IMMEDIATE or ALTER TABLESPACE OFFLINE TEMPORARY statement require media recovery.

After the upgrade, tablespaces that are offline when you open the new Oracle9i database remain in Oracle7 database file format. The offline tablespaces can be brought online at any time after the upgrade, and the file headers are converted to Oracle9i 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 perform another upgrade.

6. Make sure no user or role has the name `MIGRATE`, because the `MIG` 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.

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

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 the upgrade is complete.

See Also: The troubleshooting information in "[OPTIMAL Setting for the SYSTEM Rollback Segment](#)" on page D-26.

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 complete the upgrade successfully. If there is not enough space in your SYSTEM rollback segment, then you may encounter errors when you run the MIG utility.

9. 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 Database Globalization Support Guide* for information about setting NLS_LANG

10. Make sure you have DBA privileges, which are required to run the MIG 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.

11. Make sure no other DBA with `RESTRICTED SESSION` privilege connects to the database while the MIG utility is running. Also, "Normal" users should not connect to the database during an upgrade.
12. Shut down the Oracle7 database cleanly using the `SHUTDOWN NORMAL` or `SHUTDOWN IMMEDIATE` commands; do *not* use `SHUTDOWN ABORT`. The Oracle7 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 shut down all instances.

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

Review MIG Utility Command-Line Options

The next task in the upgrade process is running the Oracle9i MIG utility. Before you begin that task, review the following MIG utility command-line options because you may need to specify some of them when you run the MIG utility. In addition, your operating system-specific Oracle documentation may contain more information about MIG utility command-line options.

<code>CHECK_ONLY</code>	When <code>true</code> , the MIG utility performs space use calculations without performing an upgrade. When <code>false</code> , the MIG utility performs both space usage calculations and the upgrade. This command-line option is mutually exclusive with <code>NO_SPACE_CHECK</code> .
<code>DBNAME</code>	Specifies the name of the database to upgrade (<code>DB_NAME</code> in the initialization parameter file).
<code>MULTIPLIER</code>	Specifies the initial size of the Oracle9i <code>i_file#_block#</code> index relative to the Oracle7 <code>i_file#_block#</code> index. For example, <code>MULTIPLIER=30</code> triples the initial size when the index is created. If no <code>MULTIPLIER</code> command-line option is specified, then the MIG utility uses the <code>i_file#_block#</code> value of 15, creating an index for Oracle9i that is 1.5 times larger than the Oracle7 <code>i_file#_block#</code> index.

<code>NEW_DBNAME</code>	Specifies a new name for the upgraded database. The default name "DEFAULT" should not be used; choose a more meaningful name.
<code>NO_SPACE_CHECK</code>	When <code>true</code> , the MIG utility does not perform a space usage check before the upgrade. When <code>false</code> , the MIG utility performs a space usage check before the upgrade. This command-line option is mutually exclusive with <code>CHECK_ONLY</code> .
<code>PFILE</code>	Specifies the name of the initialization parameter file. If no <code>PFILE</code> command-line option is specified, then the MIG utility uses the default initialization parameter file. Note: On UNIX, the pathname must be enclosed by double-quotes escaped by a backslash, for example: <pre>mig PFILE=\" /tmp/mig/pfile\"</pre>
<code>SPOOL</code>	Specifies the filename for the spool output. Note: On UNIX, the pathname must be enclosed by double-quotes escaped by a backslash, for example: <pre>mig SPOOL=\" /tmp/mig/spool\"</pre>

Run the MIG Utility

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

- [Run the MIG Utility on UNIX Operating Systems](#)
- [Run the MIG Utility on Windows Platforms](#)

Run the MIG Utility on UNIX Operating Systems

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

1. At a command prompt, change to the `ORACLE_HOME/bin` directory in your release 9.2 installation.
2. Run the MIGPREP utility.

The MIGPREP utility prepares the Oracle7 environment for upgrading by copying required files from the Oracle9i Oracle home to the Oracle7 Oracle home. Enter the following to run MIGPREP:

```
migprep new_oracle_home old_oracle_home
```

Where *new_oracle_home* is the complete path of the new Oracle9i Oracle home directory and *old_oracle_home* is the complete path of the old Oracle7 Oracle home directory.

For example, if your new Oracle9i Oracle home is `/oracle/product/9.2` and your old Oracle7 Oracle home is `/oracle/product/7.3`, then enter the following:

```
migprep /oracle/product/9.2 /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
```

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

A common upgrade problem is running out of space in the SYSTEM tablespace during an upgrade. The MIG utility will not complete the upgrade unless sufficient space is allocated in the SYSTEM tablespace. To determine disk space requirements for a successful upgrade, run the MIG 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 MIG utility to assess the amount of disk space required for the upgrade, 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 MIG utility does not build the Oracle9i data dictionary or perform any other upgrade 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 upgrade script that you will run later in the upgrade process.

6. Enter the following to run the MIG utility:

```
mig
```

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

See Also: ["Review MIG Utility Command-Line Options"](#) on page D-9 for information about command-line options. Oracle Corporation recommends using the `SPOOL` option, because it makes it easier to check your results when the upgrade is complete.

Run the MIG Utility on Windows Platforms

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

1. In the new Oracle9i Oracle home, run the MIG utility by entering the MIG utility command at a command 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 just the `PFILE` option to run with a default set of options, or you can enter `mig` followed by more selected options.

See Also: ["Review MIG Utility Command-Line Options"](#) on page D-9 for information about command-line options. Oracle Corporation recommends using the `SPOOL` option, because it makes it easier to check your results when the upgrade is complete.

2. If the *Oracle7 Password* appears when you run the MIG 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.

Check the MIG Utility Results

Check the results after running the MIG utility. The MIG utility generates informational messages and echoes its progress as it runs the `migrate.bsq` script. If the MIG utility exits with an `ORA-` error, then check ["Troubleshooting MIG Utility Errors"](#) on page D-24 for information about the error and the actions to perform to resolve the problem.

The MIG utility creates a convert file that contains the information of the Oracle7 control file. Later in the upgrade process, the convert file is used by `ALTER DATABASE CONVERT` to create a new control file in Oracle9i.

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 `convsid.dbf`, where `sid` is your Oracle7 instance ID. On Windows platforms, the default location is `ORACLE_HOME\rdbms` in the Oracle9i environment, and the default filename in this directory is `convert.ora`.

Caution: Do not open the Oracle7 database, which was shut down by the Oracle9i MIG 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 Oracle9i. If the Oracle7 database is opened after running the MIG utility, then the SCN check will fail when the database is converted to Oracle9i, and the following error will be displayed:

```
ORA-01211: Oracle7 datafile is not from migration
to Oracle8
```

Therefore, if the Oracle7 database is opened, then you must rerun the MIG utility.

Preserve the Oracle7 Database

After you successfully run the MIG 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 in Oracle9i, then you can restore the backup, start the Oracle7 database, and complete the procedure in "[Abandoning the Oracle7 Upgrade](#)" on page D-31.
- It can be used as the first Oracle9i backup for an Oracle9i recovery.
- If an error occurs at Oracle9i database convert time (`ALTER DATABASE CONVERT` or `ALTER DATABASE OPEN RESETLOGS MIGRATE`), 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 MIG utility, then you must rerun the MIG 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

MIG Utility Messages

The MIG utility may return error messages and informational messages during the upgrade process. This section describes errors you may encounter when using the MIG utility. For each error, a description of its probable causes and instructions for

corrective action are provided. Informational messages are also listed, but they require no corrective action.

If you are using the Database Upgrade Assistant, then the MIG utility messages are recorded in a log file. See the online help for the Database Upgrade Assistant for information about accessing its log files. Also, if you are using the Database Upgrade Assistant and the recommended action for a message includes rerunning the MIG utility, then you should rerun the Database Upgrade Assistant.

The following messages are listed in alphabetical order:

cannot reduce file number bits in DBA during migration

Cause: The MIG 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 MIG 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 MIG utility.

Action: Rerun the MIG utility using only one of these options.

client nls_character set does not match server nls_character set - 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: A command-line argument was entered 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: A command-line argument was used 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 MIG utility, of `migrate.bsq`, and of the Oracle9i software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems and rerun the MIG 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 MIG utility, of `migrate.bsq`, and of the Oracle9i software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems and rerun the MIG 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 MIG utility, of `migrate.bsq`, and of the Oracle9i software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems and rerun the MIG utility.

could not find a single contiguous extent of *number* bytes for c_file#_block#

Cause: Your `SYSTEM` tablespace does not have enough contiguous space.

Action: Add free space to your `SYSTEM` tablespace and rerun the MIG utility.

could not find a single contiguous extent of *number* bytes for *c_ts#*

Cause: Your `SYSTEM` tablespace does not have enough contiguous space.

Action: Add free space to your `SYSTEM` tablespace and rerun the MIG utility.

could not find a single contiguous extent of *number* bytes for *i_file#_block#*

Cause: Your `SYSTEM` tablespace does not have enough contiguous space.

Action: Add free space to your `SYSTEM` tablespace and rerun the MIG utility.

could not find a single contiguous extent of *number* bytes for *i_ts#*

Cause: Your `SYSTEM` tablespace does not have enough contiguous space.

Action: Add free space to your `SYSTEM` tablespace and rerun the MIG utility.

could not translate logical name *string*

Cause: An internal error has occurred.

Action: Check that the logical name is defined correctly and rerun the MIG utility.

current version: *string* -- database must be release 7.1 or later

Cause: The current database is a release earlier than release 7.1.

Action: Migrate the current database to a release supported by the MIG utility on your operating system. Then, rerun the MIG utility. See your operating system-specific Oracle documentation for information about the releases supported by the MIG 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 MIG utility, of `migrate.bsq`, and of the Oracle9i software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems and rerun the MIG utility.

datafile is found in inconsistent state (internal error) -- *string*

Cause: An internal error occurred; a datafile was found in an inconsistent state.

Action: Contact Oracle Support Services.

datafile is offline while tablespace is online - apply media recovery and bring datafile online before migration -- *string*

Cause: The datafile in a tablespace is offline while the tablespace is online. The MIG utility 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 MIG utility.

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 - string

Cause: An internal error has occurred; the `migrate.bsq` script may be corrupted.

Action: Check that the version of the MIG utility, of `migrate.bsq`, and of the Oracle9i software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems and rerun the MIG 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$_CALL`, then push all deferred transactions until `SYSTEM.DEF$_CALL` is empty. If entries are in `SYSTEM.DEF$_ERROR`, then resolve and re-execute any errors in the local queue until it is empty. Rerun the MIG utility.

error calling slgtd

Cause: Error in getting current time from `slgtd`, an internal error. The MIG utility may be corrupted.

Action: Check that the version of the MIG utility, of `migrate.bsq`, and of the Oracle9i software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems and rerun the MIG 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 MIG utility.

estimated space requirement for *string* is *number* blocks

Cause: In this informational message, the MIG utility displays the space required for the object.

Action: No user action is required.

file *number* is too large for DBA conversion

Cause: An internal error has occurred; the specified file 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 MIG 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 MIG 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 MIG utility, of `migrate.bsq`, and of the Oracle9i software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems and rerun the MIG 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 required.

free space found: *number*

Cause: This informational message shows the amount of free space in the SYSTEM tablespace.

Action: No user action is required.

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 MIG 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 MIG 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 MIG utility.

migration can't proceed - database blocksize *number* is less than Oracle9i's minimum block size 2 KB

Cause: The existing database blocksize is less than 2 KB.

Action: Make sure the block size of the Oracle7 database is at least 2 KB. You may consider rebuilding the Oracle7 database. Then, rerun the MIG utility.

migration can't proceed with datafile online while tablespace offline -- *string*

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. Then, rerun the MIG utility.

migration cannot proceed with active transactions or offline tablespaces with outstanding undo

Cause: One or more tablespaces were offline with outstanding save undo when the MIG utility attempted to upgrade the database.

Action: See Step 5 on page D-6 and make sure all offline tablespaces have been taken offline cleanly. Then, rerun the MIG utility.

mounting database ...

Cause: This is an informational message. The MIG 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 MIG 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 Oracle9i *i_file#_block#* in the command line, is less than 2.

Action: Change the MULTIPLIER value to be greater than or equal to 2, and rerun the MIG utility.

NEW_DBNAME *string* too long - maximum length is 8 characters

Cause: The specified new database name is more than 8 characters long.

Action: Change the specified name for the new database to 8 or fewer characters, and rerun the MIG 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 character set 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 MIG utility with this option.

***string* 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 MIG 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 MIG utility.

ORA-number:

Cause: The MIG 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 initialization parameter file, possibly by removing any obsolete parameters. Then, rerun the MIG utility.

parameter file exceeds *number* bytes

Cause: The initialization parameter file for your Oracle7 database exceeds the maximum size.

Action: If possible, reduce the size of your initialization 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 *string*

Cause: An internal error has occurred reading the specified file.

Action: Make sure the file and disk are not corrupted. Fix any corruption before you rerun the MIG utility.

short read, *number* bytes requested, *number* bytes read

Cause: There was a problem reading the control file.

Action: Check the control file for corruption, fix any problems, and rerun the MIG utility.

shut down database (abort) ...

Cause: An internal error has occurred.

Action: Additional error messages should inform you of the cause of the shut-down. Follow the actions suggested for these additional messages.

shutting down database ...

Cause: This is an informational message. The MIG 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 MIG 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 MIG utility exceeds the maximum size.

Action: Shorten the string in the command-line argument and rerun the MIG utility.

Tablespace of datafile not taken offline normal. Bring tablespace online, offline normal, or drop before migration -- *string*

Cause: A tablespace was taken offline using IMMEDIATE or TEMPORARY.

Action: Either bring the tablespace online and then take it offline using NORMAL, or drop the tablespace. Then, rerun the MIG utility.

too many args in command (*number max*)

Cause: Too many arguments were specified 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 MIG 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 and rerun the MIG utility.

unable to open file *string*

Cause: An internal error has occurred, or a file was not in the expected location, when you started the MIG 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 reside, are not corrupt. Fix any problems and rerun the MIG utility.

unable to read file *string*

Cause: An internal error has occurred or a file was not in the expected location when you started the MIG 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 reside, are not corrupt. Fix any problems and rerun the MIG utility.

unable to write file *string*

Cause: An internal error has 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 MIG 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 successfully run the MIG utility.

Action: Make sure you have the specified amount of additional space before running the MIG utility.

Troubleshooting MIG Utility Errors

Errors may be caused by the following actions or omissions:

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

Problems Using the MIG Utility

General upgrade problems may occur when you run the MIG utility, but they are caused by your database system's configuration. While the MIG utility is performing the necessary actions to upgrade the Oracle7 database, an error is generated by your Oracle software. Typically, when such an error occurs, the MIG utility stops and displays one or more error messages.

If you encounter one of the following problems when you run the MIG utility, then perform the suggested actions, and then rerun the MIG utility.

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 upgrade.

It is also possible to run out of space in the temporary tablespace during the upgrade. If you do, then add a new datafile to the temporary tablespace and allocate enough space to the new datafile to successfully complete the upgrade.

See Also: ["Space Requirements"](#) on page D-3 and Step 5 on page D-11 for 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 MIG utility.

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 upgrade. You must ensure that the `SYSTEM` rollback segment is large enough for the upgrade to complete successfully.

Both the MIG utility and the Database Upgrade 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 MIG utility or the Database Upgrade Assistant handles the space management tables.

If the `SYSTEM` rollback segment has an `OPTIMAL` setting, then extents are deallocated dynamically when their data is no longer needed for active transactions. The dynamic deallocation 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 the upgrade.

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 MIG utility, then see Step 7 on page D-7 for information on 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 MIG 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 MIG 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 MIG utility in the background by using the Database Upgrade Assistant, then restore the backup of the database being upgraded and then rerun the Database Upgrade Assistant. Choose the Custom migration option in the Database Upgrade Assistant. When you are prompted for the `MULTIPLIER` value, enter a value greater than the default of 15.

See Also: ["Review MIG Utility Command-Line Options"](#) on page D-9 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 upgrade process after you run the MIG 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 issue the `ALTER DATABASE CONVERT` statement, then perform the suggested actions to correct the problem.

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 b on page 3-15

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: 'name'
```

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 b on page 3-15 and Step 11 on page 3-21

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 (*convsid.dbf* on UNIX and *convert.ora* on Windows platforms) generated by the MIG utility was not found in the expected location. On UNIX, the expected location is the *ORACLE_HOME/dbs* directory in the Oracle9i environment; on Windows platforms, the expected location is the *ORACLE_HOME\rdbms* directory in the Oracle9i environment. The convert file must be moved to this location before you issue the ALTER DATABASE CONVERT statement.

See Also: Step 3 on page 3-17

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 Oracle9i environment; on Windows platforms, the correct directory is *ORACLE_HOME\database* in the Oracle9i environment.
- The *REMOTE_LOGIN_PASSWORDFILE* initialization parameter is set to *EXCLUSIVE* in the initialization parameter file.

To continue with the upgrade, complete the following steps:

1. Shut down 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 MIGRATE` statement:

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

5. Continue with the upgrade process starting with Step 12 on page 3-22.

You cannot use the existing password file because it is no longer valid. If you want to use a password file with Oracle9i, then re-create 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 'name' in controlfile is not 'name'
```

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 `convsid.dbf` filename.
- The Oracle9i instance ID set by the `ORACLE_SID` environment variable does not match the database name in the `convsid.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 `convsid.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 `convsid.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 upgrade process by issuing the `ALTER DATABASE OPEN RESETLOGS MIGRATE` statement. However, if you encounter problems, then restore the backup you created before you started the upgrade process, and use it to start the upgrade again from the beginning. Start at the beginning of [Chapter 3](#), but make sure you performed the pre-upgrade actions described in [Chapter 2](#) and in this appendix.

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 MIG 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 MIG utility, or the database was opened by Oracle7 after you ran the MIG utility. Only the datafiles that were current when the MIG utility ran can be accessed by Oracle9i.

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 Oracle9i. If the database is opened under Oracle7 after the MIG 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. Shut down 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 MIG utility completes. Start from the beginning of [Chapter 3](#).

Abandoning the Oracle7 Upgrade

If you performed a backup of your Oracle7 database *before* running the MIG utility, then the easiest way to abandon an upgrade is to restore that backup. However, if you do not have a backup, or if you made the backup after running the MIG utility, then you must complete the procedure described in this section to abandon the upgrade.

You can run the Oracle9i MIG utility multiple times and still return to the Oracle7 database. However, running the MIG utility automatically eliminates the Oracle7 database catalog views. Therefore, to return to the Oracle7 database after running the MIG utility, you must run the Oracle7 `catalog.sql` script to restore the Oracle7 database catalog views.

Note: You cannot use the following procedure to abandon the upgrade if you have already executed the `ALTER DATABASE CONVERT` statement. If you have executed this statement and want to return to Oracle7, then complete the downgrade procedure in [Chapter 8, "Database Migration Using Export/Import"](#).

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

1. Start the Oracle7 database using Server Manager.

2. Drop the MIGRATE user:

```
DROP USER MIGRATE CASCADE;
```

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

```
@catalog.sql  
@catproc.sql
```

4. Run `catsvrmg.sql`:

```
@catsvrmg.sql
```

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

```
@catparr.sql
```

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

```
@catrep.sql
```

Migration Issues for Physical Rowids

Release 8.0 introduced new internal and external formats for physical rowids 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 section includes the following topics:

- [Upgrading Applications and Migrating Data](#)
- [The DBMS_ROWID Package](#)
- [Snapshot Refresh](#)
- [Oracle7 Client Compatibility Issues](#)
- [ROWID Migration and Compatibility Issues](#)

Note: In the rest of this section, references to new rowids include rowid functionality that was introduced in release 8.0. Also, the word "rowid" means "physical rowid". This appendix does not discuss the UROWID (universal rowid) datatype. See [Chapter 5, "Compatibility and Interoperability"](#) for compatibility issues relating to the UROWID datatype.

Upgrading Applications and Migrating Data

Rowids can be stored in columns of ROWID datatype and in columns of character type. Stored Oracle7 rowids become invalid after an upgrade to Oracle9i. Therefore, stored Oracle7 rowids must be converted to the new format.

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 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.

The columns that contain rowid values (in ROWID datatype format or in character format) must be migrated if they point to tables that were upgraded to Oracle9i. 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 upgraded tables still point to Oracle7 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 the upgrade to Oracle9i. 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 Oracle9i. You can upgrade different tables at different times or multiple tables in parallel. Make sure the upgrade is done *before* the Oracle7 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 `DBMS_ROWID` functionality, to migrate stored rowids from Oracle7 format to the new format.

Data migration by the MIG utility or the Database Upgrade Assistant applies only to rowids stored in a user-defined column. All system-stored rowids (such as in indexes) remain valid after the upgrade, and do not require specific actions to be migrated. Also, indexes are not invalidated because, during the upgrade to Oracle9i, indexes can continue to use the restricted ROWID datatype format.

Note: Importing a column containing rowids should produce a message indicating that special attention might be required to re-establish the validity of the rowids. Special attention is necessary for *all* rowids being imported. Thus, database 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 package contains the following functionality:

- Creation and interpretation of rowids in both the Oracle7 format and in the new format
- Conversion between Oracle7 rowids and new 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 VARCHAR2 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 VARCHAR2 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 new (extended) format.
- `ROWID_TO_RESTRICTED` converts a rowid from the new (extended) format to the Oracle7 (restricted) format.
- `ROWID_VERIFY` checks whether a given rowid can be converted from Oracle7 format to the new 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 D-34 for more information

`ROWID_TO_EXTENDED` returns a new (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 are 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, $\langle n \rangle 0 . \langle m \rangle 0 . \langle p \rangle 0$ is supplied for rowid, then the table name is ignored and a restricted rowid of the form 00000000.0000.0000 is returned.
- If a new 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. 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 new ROWID datatype format forces all rowid snapshots to perform a complete refresh when both master and snapshot sites are upgraded to Oracle9i.

See Also: [Appendix E, "Database Migration and Compatibility for Replication Environments"](#) for more information about replication compatibility

Oracle7 Client Compatibility Issues

Oracle7 clients can access a release 8.0 or higher database, and release 8.0 and higher clients can access an Oracle7 database. Binary and character values of the pseudo column ROWID and of columns of datatype ROWID that are returned by

an Oracle7 database to a release 8.0 and higher database are always in restricted format, because Oracle7 cannot recognize the extended format ROWID.

The `DBMS_ROWID` package can be used for interpreting the contents of Oracle7 rowids and for creating the rowids in Oracle7 format.

An Oracle7 client accessing a release 8.0 and higher database receives the rowid in the new (extended) format. Therefore, the client cannot interpret the contents of the new rowids.

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 a release 8.0 and higher database. Similarly, a release 8.0 and higher 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 a Release 8.0 and Higher Client

The ROWID values that are returned are always restricted ROWIDs. Also, ROWID values returned to an Oracle7 server are always restricted ROWIDs.

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 a Release 8.0 or Higher Database from an Oracle7 Client

Release 8.0 and higher 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.

Export/Import

It is not possible for an Oracle7 client to import a release 8.0 or higher table that has a ROWID column (not the ROWID pseudocolumn), if any row of the table contains an extended ROWID value.

Changes to Initialization Parameters and the Data Dictionary in Release 8.0

This section lists changes to initialization parameters and the data dictionary in release 8.0.

Initialization Parameter Changes in Release 8.0

The following sections list changes to initialization parameters in release 8.0.

Renamed Initialization Parameters in Release 8.0

The initialization parameters listed in [Table D-1](#) were renamed in release 8.0:

Table D-1 Initialization Parameters Renamed in Release 8.0

Pre-Release 8.0 Name	Release 8.0 and Higher Name
ASYNC_READ	DISK_ASYNC_IO
ASYNC_WRITE	DISK_ASYNC_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 Release 8.0

The following initialization parameters were made obsolete in release 8.0:

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 9.2 database using one or more of these obsolete initialization parameters will result in an error, and the database will not start.

Static Data Dictionary View Changes in Release 8.0

The following sections list changes to static data dictionary views in release 8.0.

Static Data Dictionary Views Obsolete in Release 8.0

The following static data dictionary views were made obsolete in release 8.0:

ALL_HISTOGRAMS	DBA_HISTOGRAMS
DEFCALL	USER_HISTOGRAMS

Database Migration and Compatibility for Replication Environments

This appendix describes the steps that you must complete to upgrade a replication environment from Oracle7 to Oracle9i. This appendix covers the following topics:

- [Database Migration Overview for Replication](#)
- [Upgrading All Sites at Once](#)
- [Upgrading Incrementally](#)
- [Upgrading to Primary Key Materialized Views](#)
- [Features Requiring an Upgrade to a Higher Release of Oracle](#)
- [Obsolete Procedures](#)

Note: This appendix addresses upgrading from Oracle7 to Oracle9i. It does not address upgrading from release 8.0 or higher to Oracle9i. For more information about upgrading from release 8.0 or higher to the current Oracle9i release, see [Chapter 3, "Upgrading a Database to the New Oracle9i Release"](#).

Database Migration Overview for Replication

In some cases, you may find it easiest to upgrade your replication environment, particularly the multimaster component of your environment, in one step. Typically, this type of upgrade is only possible for small configurations. If you have a large configuration, then you might consider upgrading 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 upgrade methods are supported for replication environments:

- Manual Upgrade
- Database Upgrade Assistant
- Full database export from Oracle7 and import to Oracle9*i*

After upgrading a master site to Oracle9*i*, perform a complete refresh of all 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 upgraded 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 upgraded 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 Oracle9i replication features require that all sites be successfully upgraded to Oracle8i or higher before the features can be used, and certain Oracle9i replication features require that all sites be successfully upgraded to Oracle9i before the features can be used. For example, to replicate objects based on user-defined types, all sites must be Oracle9i. These features are listed in ["Features Requiring an Upgrade to a Higher Release of Oracle"](#) on page E-18.

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 Oracle9i
- 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*.

Upgrading All Sites at Once

This section describes upgrading all master sites in your multimaster environment to Oracle9i at once. Any materialized view sites that you do not also upgrade to Oracle9i must be upgraded to Oracle7 release 7.3.4 or higher. If you want to upgrade your sites incrementally instead, see ["Upgrading Incrementally"](#) on page E-6.

Complete the following steps to upgrade all master sites and (optionally) materialized view sites at once:

1. Stop all propagation and refreshing from materialized view sites to all masters that you are upgrading. 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 the `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. Upgrade all master sites using one of the upgrade methods discussed in [Chapter 3](#).

Alternatively, you can use Export/Import. To export a full database from Oracle7 release 7.3.3 or higher and import to Oracle9i, 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 Oracle9i database using the Oracle9i Import utility under the `SYSTEM` schema with `FULL=y`.

You may also export data from individual Oracle7 tables, import the data to Oracle9i tables, and then configure those tables as masters in an Oracle9i replication environment using standard replication procedures.

If you use export/import, then you may need to drop and re-create the materialized views that are based on the master tables.

See Also:

- [Chapter 8, "Database Migration 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 upgrade these sites to Oracle9i. Upgrading these materialized view sites to Oracle9i is preferable.

See Also:

- Your Oracle7 documentation for information about upgrading your materialized view sites to release 7.3.4
 - ["Incremental Upgrade of Materialized View Sites"](#) on page E-8 for instructions on upgrading your materialized view sites to Oracle9i
9. Perform a complete refresh on all materialized views at all materialized view sites that have master sites upgraded to Oracle9i. Before the refresh, make sure you have "unbroken" any jobs that you may have "broken" during the upgrade 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 upgrade all of a master's materialized view sites to Oracle9i when the master site is upgraded to Oracle9i (that is, you do not need to upgrade the materialized view sites incrementally), then you can alternatively drop the materialized view logs for the master and re-create 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 E-15 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 Oracle9i environment. Unnecessary privileges may also be revoked. Be careful not to drop accounts that are needed to maintain any Oracle7 materialized view sites.

Upgrading Incrementally

It is possible to incrementally upgrade your replication environment. However, you must carefully analyze the interdependencies between sites to ensure that they can continue to interoperate throughout the upgrade. [Table E-1](#) describes the conditions that must be met to allow Oracle7 and Oracle9i replication sites to interoperate.

Table E-1 Interoperability in a Replication Environment

Environment	Action	Condition
Multimaster	Upgrade master site from Oracle7 to Oracle9i.	All other master sites must be Oracle7 release 7.3.3 or higher.

Table E-1 (Cont.) Interoperability in a Replication Environment

Environment	Action	Condition
Master with dependent materialized views	Upgrade master site from Oracle7 to Oracle9i.	All dependent materialized view sites must be Oracle7 release 7.3.4 or higher.
Master with dependent materialized views	Upgrade materialized view site from Oracle7 to Oracle9i.	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 upgrade, 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 an Incremental Upgrade"](#) on page E-7 to prepare your Oracle7 master sites for an incremental upgrade.
2. Incrementally upgrade all materialized view sites to Oracle9i by completing the steps in ["Incremental Upgrade of Materialized View Sites"](#) on page E-8.
3. Incrementally upgrade all master sites to Oracle9i by completing the steps in ["Incremental Upgrade of Master Sites"](#) on page E-10.

See Also: Your Oracle7 documentation for information about upgrading your Oracle7 sites to release 7.3.3 or higher

Preparing Oracle7 Master Sites for an Incremental Upgrade

Before beginning an incremental upgrade 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 Oracle9i 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 Upgrade of Materialized View Sites

Before you can upgrade a materialized view site to Oracle9i, 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 an incremental upgrade.

See Also:

- Your Oracle7 documentation for information about upgrading your Oracle7 sites to release 7.3.3 or higher
- ["Preparing Oracle7 Master Sites for an Incremental Upgrade"](#) on page E-7 for information about preparing your master sites

To incrementally upgrade your Oracle7 materialized view sites to Oracle9i, 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. Perform one of the upgrade methods discussed in [Chapter 3](#).

Alternatively, you can use Export/Import. To export a full database from Oracle7 release 7.3.3 or higher and import to Oracle9i, 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 Oracle9i database using the Oracle9i Import utility under the `SYSTEM` schema with `FULL=Y`.

You may also export data from individual Oracle7 tables, import the data to Oracle9i tables, and then configure those tables as masters in an Oracle9i replication environment using standard replication procedures.

See Also:

- [Chapter 8, "Database Migration 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 Oracle9i 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 upgraded, according to your schedule.

Incremental Upgrade of Master Sites

Before upgrading a master site from Oracle7 to Oracle9i, 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 an Incremental Upgrade](#)" on page E-7.
- 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 upgrade your Oracle7 master sites to Oracle9i, complete the following steps:

1. Pick a master site to upgrade. You should upgrade 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 upgraded.

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 upgraded.

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. Upgrade the master site using one of the upgrade methods discussed in [Chapter 3](#).

Alternatively, you can use Export/Import. To export a full database from Oracle7 release 7.3.3 or higher and import to Oracle9i, 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 Oracle9i database using the Oracle9i Import utility under the `SYSTEM` schema with `FULL=y`.

You may also export data from individual Oracle7 tables, import the data to Oracle9i tables, and then configure those tables as masters in an Oracle9i replication environment using standard replication procedures.

If you use export/import, then you may need to drop and re-create the materialized views that are based on the master tables.

See Also:

- [Chapter 8, "Database Migration 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 Oracle9i master sites should already exist if you prepared your Oracle 7 master site for compatibility with Oracle9i using the directions in "[Preparing Oracle7 Master Sites for an Incremental Upgrade](#)" on page E-7.

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 upgraded to Oracle9i (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 uncheduling 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 upgraded to Oracle9i. Because of the new rowid format introduced in Oracle8, all the materialized view logs of master tables are truncated during the upgrade.
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 upgrade all of the master's materialized view sites to Oracle9i when the master site is upgraded to Oracle9i (that is, you do not need to upgrade the materialized view sites incrementally), then you can alternatively drop the materialized view logs for the master and re-create 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 E-15 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 Oracle9i 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 upgraded, according to your schedule.

Upgrading to Primary Key Materialized Views

When a materialized view site and its master have been upgraded to Oracle9i, you can upgrade your rowid materialized views to Oracle9i 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 Oracle9i materialized view sites by altering the materialized views to convert them to primary key materialized views. Oracle9i masters that have been altered to log primary key as well as rowid information can support Oracle7 rowid materialized views as well as Oracle9i rowid and primary key materialized views simultaneously to allow for an incremental upgrade.

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 Oracle9i 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 Oracle9i master sites have been configured to support primary key materialized views, complete the following steps at the Oracle9i 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 re-create 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 your master site in Step 1, then you need to "unbreak" your jobs to resume your replication activity from your materialized view sites.

Features Requiring an Upgrade to a Higher Release of Oracle

Oracle adds new features to each major release of the Oracle database server. The following sections list the features that can only be used if you upgrade your database to a higher release of Oracle.

See Also: The "What's New in Replication" section of *Oracle9i Replication* for more information about these new replication features

Features Requiring Oracle9i

All replication sites involved must be running Oracle9i 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 Oracle8i or Higher

Master sites must be running Oracle8i 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 Oracle8i 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 Oracle9i 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`

Numerics

- 2 GB and larger files
 - operating system dependencies, 5-51
- 32-bit to 64-bit conversion. *See* word size

A

- Ada. *See* SQL*Module for Ada
 - Advanced Queuing
 - compatibility, 5-37
 - privileges, 5-37
 - rule based subscriptions, 5-38
 - interoperability, 5-37
 - sender's ID column, 5-38
 - upgrading, 4-17
 - AL24UTFSS character set
 - desupported in Oracle9i, 5-33
 - ALTER DATABASE CONVERT statement, 3-21, D-2
 - rerunning, D-30
 - ALTER DATABASE OPEN RESETLOGS MIGRATE statement, 3-22, D-3
 - ALTER DATABASE RESET COMPATIBILITY statement, 5-9, 5-10
 - ALTER DATABASE RESET LOGS statement, D-5
 - ALTER TABLE statement
 - bitmap index invalidation, 5-30
 - ANALYZE TABLE VALIDATE STRUCTURE statement
 - change in release 8.1, 5-21
 - applications
 - client/server configurations
 - upgrading, 6-3
 - compatibility, 5-19, 6-2
 - development
 - role during the upgrade, 1-8
 - index-organized tables
 - compatibility, 5-20
 - interoperability, 5-19
 - linking with newer libraries, 6-6
 - OCI
 - compatibility, 5-21
 - interoperability, 5-21
 - physical ROWIDs and UROWIDs, 5-20
 - PL/SQL
 - compatibility, 5-23
 - interoperability, 5-23
 - precompiler
 - compatibility, 5-22
 - interoperability, 5-22
 - running against older server, 6-5
 - upgrading, 6-1
 - compatibility rules, 6-4
 - options, 6-6
 - relinking rules, 6-4
- AQ. *See* Advanced Queuing
- archive log destination parameters
 - new in release 8.1, A-11
 - archived redo logs
 - analyzing
 - from other databases, 5-47
 - compatibility, 5-46
 - rearchiving, 5-46
 - archiving
 - error detection behavior, 5-46
 - AUDIT_TRAIL initialization parameter
 - for upgrading, D-25

- automatic segment-space managed tablespaces
 - change in compatibility level, 5-15
- automatic segment-space managed tablespaces with LOBs
 - downgrading, 7-7

B

- backups
 - after upgrading, 4-2
 - before upgrading, D-14
 - compatibility, 5-43
 - EBU, 5-44
 - preparing a strategy, 2-8
 - Recovery Manager and EBU, 5-44
- backward compatibility
 - of ROWIDs, D-38
- bitmap indexes
 - invalidations, 5-30
 - during upgrade, 4-20
- bitmap secondary indexes
 - dropping from index-organized tables, 7-9
- blocks
 - block size
 - DB_BLOCK_SIZE initialization parameter, D-4
 - minimums for upgrading, D-4

C

- CATALOG5.SQL script
 - obsolete, 5-30
- CATALOG6.SQL script
 - obsolete, 5-30
- CATALOG.SQL script, 3-23, 7-28, D-3
- CATCLUST.SQL script, 7-28
- CATEXP7.SQL script, 8-2
- CATPARR.SQL script, 7-28
- CATPROC.SQL script, 3-23, 7-28, D-3
- CATREP.SQL script, 7-28
- change passwords
 - for oracle-supplied accounts, 4-2
- changes to initialization parameters, D-39
- changes to static data dictionary views, D-40
- CHAR

- maximum size
 - change, 5-19
- CHARACTER keyword
 - behavior differences, 5-26
- character sets
 - upgrading the database, 3-10
 - varying-width
 - CLOBs and NCLOBs, 5-31
- CHECK_ONLY
 - MIG utility option, D-9
- ckpdom.ora file, B-19
- ckptop.ora file, B-19
- client-server configurations, 1-9
- CLOBs
 - compatibility, 5-31
- collections
 - collection columns
 - user-specified storage, 5-35
 - collection locators
 - compatibility, 5-35
- command line
 - command-line options
 - MIG utility, D-9
- commands
 - COMMIT FORCE, D-5
 - ROLLBACK FORCE, D-5
- comments
 - differences between Server Manager and SQL*Plus, C-7
- COMMIT command
 - differences between Server Manager and SQL*Plus, C-14
- COMMIT FORCE command, D-5
- COMMIT keyword
 - behavior differences, 5-26
- compatibility, 5-1
 - Advanced Queuing, 5-37
 - ALTER DATABASE RESET COMPATIBILITY statement, 5-9
 - ANALYZE VALIDATE STRUCTURE statement, 5-21
 - applications, 5-19, 6-2
 - index-organized tables, 5-20
 - physical ROWIDs and UROWIDs, 5-20

- archived redo logs, 5-46
- automatic segment-space managed
 - tablespaces, 5-15
- backup, 5-43
- change in maximum size
 - CHAR, 5-19
 - RAW, 5-19
 - VARCHAR2, 5-19
- checking level for features, 5-10
- compatibility level, 5-6
 - features requiring, 5-10
- COMPATIBLE initialization parameter, 5-2
- CREATE TABLE behavior change, 5-37
- data dictionary, 5-29
- datafiles, 5-27
- datatypes, 5-31
- date columns in dynamic performance
 - views, 5-32
- dictionary managed tablespaces, 5-15
- downgrading, 5-6
- Export/Import, 8-2
- Heterogeneous Services agents, 5-48
- initialization parameters, A-5
- LOB index clause, 5-31
- LOBs, 5-31
 - CLOBs and NCLOBs, 5-31
- LogMiner, 5-46
- materialized views, 5-47
- nested tables, 5-35
- NLS and NCHAR environment variables, 5-34
- object types, 5-16
- OCI, 5-21
 - link line, 5-21
 - thread safety, 5-21
- optimization, 5-38
- Oracle Managed Files, 5-16
- Oracle OLAP, 5-16
- PL/SQL, 5-23
 - integrated SQL analysis, 5-24
 - PLSQL_V2_COMPATIBILITY initialization parameter, 5-25
- precompilers, 5-22
- recovery, 5-43
- removing incompatibilities, 7-2
- replication, 5-48
 - resetting for database, 7-20
- ROWID issues, D-38
- rowids, 5-32
- schema objects, 5-30
- scripts
 - UTLCHN1.SQL, 5-36
 - UTLEXPT1.SQL, 5-37
- standby database, 5-45
- STARTUP, 5-27
- tablespaces, 5-27
 - user-defined datatypes, 5-34
- varrays
 - stored as LOBs, 5-35
- compatibility level
 - checking, 7-2
 - resetting, 5-10
- COMPATIBLE initialization parameter, 5-2
 - checking, 5-6
 - database structures, 5-3
 - setting, 5-7
 - features requiring, 5-10
 - when to set, 5-6
- connections
 - load balancing in Net8, 5-50
- connect-time failover
 - GLOBAL_DBNAME networking parameter in listener.ora, B-11
- ConText
 - migrating to Oracle Text, D-5
- control files
 - renaming or removing for migration, 3-16
 - renaming or removing for upgrading, 3-15
- CREATE LIBRARY command
 - differences between Server Manager and SQL*Plus, C-13
- CREATE TABLE
 - AS subquery
 - behavior change, 5-37
 - behavior change, 5-37
- CREATE TYPE command
 - differences between Server Manager and SQL*Plus, C-13

D

- data copying, 2-7
- data dictionary
 - compatibility, 5-29
 - protection, 5-30
- database
 - failure to open after removing incompatibilities, 5-10
- database administrator
 - role during the upgrade, 1-7
- database migration
 - definition, 1-3
 - overview, 1-3
 - overview for replication, E-2
 - terminology, 1-2
 - using Export/Import, 8-1
- Database Upgrade Assistant
 - advantages, 2-4
 - running, 3-4
- databases
 - backing up for upgrading, D-14
 - downgrading, 7-21
 - test upgrade results, 4-22
 - tuning after upgrading, 4-22
- datafiles
 - compatibility, 5-27
 - offline during upgrade, D-3
- datatypes
 - compatibility, 5-31
- date constraints
 - checking for bad, 4-20
- DB_BLOCK_CHECKSUM
 - new default value, A-5
- DB_BLOCK_CHECKSUM initialization parameter
 - compatibility, A-5
- DB_BLOCK_SIZE initialization parameter
 - for upgrading, D-4
- DB_DOMAIN initialization parameter, B-11
 - compatibility, A-8
- DB_NAME initialization parameter, B-11
- DBMS
 - precompiler command line option, 5-23
- DBMS_APPLICATION_INFO package
 - SET_SESSION_LONGOPS procedure
 - syntax change, 5-38
- DBMS_ROWID package, D-34
- DBMS_STATS package
 - upgrading statistics tables, 4-19
- DBNAME
 - MIG utility option, D-9
- DBUA. *See* Database Upgrade Assistant
- DEC keyword
 - behavior differences, 5-26
- definitions. *See* terminology
- DEGREE keyword
 - in PARALLEL clause, 5-39
- deinstalling, 1-12
- deprecated dynamic performance views, A-20
- deprecated features
 - dictionary managed tablespaces, 5-15
 - Oracle Dynamic Services, 5-17
 - Oracle Syndication Server, 5-17
- deprecated initialization parameters, A-2
- deprecated static data dictionary views, A-14
- Developer/2000 Applications
 - upgrading, 6-10
- dictionary managed tablespaces
 - compatibility, 5-15
 - deprecated, 5-15
 - interoperability, 5-15
- distributed databases
 - preparing to upgrade, D-5
- DML_LOCKS initialization parameter
 - compatibility, A-7
- downgrading
 - CATALOG.SQL, 7-28
 - CATPARR.SQL, 7-28
 - CATPROC.SQL, 7-28
 - Java, 7-28, 7-29
 - Messaging Gateway, 7-29
 - Oracle Workspace Manager, 7-29
 - Oracle9i Real Application Clusters, 7-28
 - ORADIM, 7-24
 - procedure for, 7-21
 - removing incompatibilities, 7-2
 - automatic segment-space managed tablespaces, 7-8
 - automatic segment-space managed tablespaces with LOBs, 7-7

- bitmap secondary indexes on index-organized tables, 7-9
- external tables, 7-9
- hash partitioned index-organized tables, 7-11
- indexes with large keys, 7-10
- LOB retention, 7-7
- non-final types, 7-19
- parallel table functions, 7-20
- partitioned index-organized tables with LOBs, 7-12
- PDML ITL invariants, 7-12
- pipelined table functions, 7-20
- release 9.0 partitioning methods, 7-10
- release 9.2 DEFAULT partitions, 7-4
- release 9.2 partitioning methods, 7-5
- release 9.2 Streams, 7-5
- release 9.2 subpartition templates, 7-6
- SQL and PL/SQL, 7-20
- subtypes, 7-19
- type evolution, 7-19
- undo tablespaces, 7-9
- user-defined aggregate functions, 7-19
- view constraints, 7-20
- replication, 7-28
- resetting database compatibility, 7-20
- scripts, 7-22
 - JVMD817.SQL, 7-28
 - JVMD901.SQL, 7-28
 - MGWD901.SQL, 7-29
 - OWMD901.PLB, 7-29
 - rerunning, 7-22
 - XMLD817.SQL, 7-29
 - XMLD901.SQL, 7-29
- view constraints, 7-20
- DUMP_TNSNAMES command, B-15
- Dynamic Discovery Option for Oracle Names migration issues, B-15
- dynamic performance views
 - changes in Oracle9i, A-20
 - date columns
 - compatibility, 5-32
 - deprecated, A-20
 - obsolete, A-22
 - with dropped columns, A-24

- with renamed columns, A-23

E

- EBU
 - backup management, 5-44
- enterprise user management
 - interoperability, 5-43
- environment variables
 - compatibility
 - NCHAR and NLS, 5-34
 - ORA_NLS32, 5-34
 - ORA_NLS33, 5-34, D-11
 - required for upgrading, 3-20
- Export utility
 - database migration, 8-1
 - requirements for upgrading, 8-3
- Export/Import
 - advantages and disadvantages, 2-6
 - benefits, 2-7
 - compatibility, 8-2
 - effects on upgraded databases, 2-6
 - incompatible data, 8-3
 - scripts
 - CATEXP7.SQL, 8-2
 - time requirements, 2-7
 - upgrade steps using, 8-3
- extended ROWIDs, D-38
- external tables
 - dropping for downgrading, 7-9

F

- FAILOVER networking parameter, B-7
- failure to open database after removing incompatibilities, 5-10
- FALSE keyword
 - behavior differences, 5-26
- fast-start parallel recovery
 - compatibility, 5-45
- fast-start rollback
 - compatibility, 5-45
- feature compatibility, 5-10
- features
 - new features, 5-10

- requiring an upgrade, E-18
- filenames
 - normalize, 4-11
- Forms
 - upgrading Oracle Forms applications, 6-10
- function-based indexes
 - invalidations
 - during upgrade, 4-16

G

- generating
 - replication support, E-5, E-13
- global database name, B-11
- GLOBAL_DBNAME networking parameter, B-11
- glossary. *See terminology*
- GREATEST_LB function
 - desupported, 5-36

H

- hash partitioned index-organized tables
 - downgrading, 7-11
- Heterogeneous Services
 - agents
 - compatibility, 5-48
 - interoperability, 5-48
 - multithreaded, 5-49

I

- identifying incompatibilities
 - UTLINCMP.SQL, 7-3
- Identix authentication, B-2
- Import utility
 - database migration, 8-1
 - requirements for upgrading, 8-3
- incompatibilities
 - removing, 7-2
- incremental upgrade, E-6
- indexes
 - bitmap, 4-20
 - function-based, 4-16
- initialization parameters
 - adjusting for Oracle9i, 3-18, 4-10

- archive log destination
 - switching to new, A-11
- changes, D-39
- changes in Oracle9i, A-2
- compatibility, A-5
 - DB_BLOCK_CHECKSUM, A-5
 - DB_DOMAIN, A-8
 - DML_LOCKS, A-7
 - JOB_QUEUE_PROCESSES, A-5
 - LOG_CHECKPOINT_TIMEOUT, A-7
 - O7_DICTIONARY_ACCESSIBILITY, A-7
 - ORACLE_TRACE_ENABLE, A-6
 - SERIALIZABLE, A-6
 - SORT_AREA_SIZE, A-6
 - SORT_DIRECT_WRITES, A-6
- COMPATIBLE, 5-2
- deprecated, A-2
- LARGE_POOL_SIZE
 - parallel execution allocation, A-8
- obsolete, A-3
- REMOTE_LOGIN_PASSWORDFILE, 3-19
- renamed, D-39
- SHARED_POOL_SIZE
 - parallel execution allocation, A-8
- INIT.ORA parameters. *See initialization parameters*
- installation
 - Oracle9i software, 3-2, 8-4
- INSTANCE_NAME initialization parameter, B-8
- INSTANCES keyword
 - removed from PARALLEL clause, 5-39
- INT keyword
 - behavior differences, 5-26
- interoperability, 5-1, 5-13
 - Advanced Queuing, 5-37
 - applications, 5-19
 - dictionary managed tablespaces, 5-15
 - Heterogeneous Services agents, 5-48
 - native dynamic SQL, 5-36
 - object types, 5-16
- OCI, 5-21
 - Oracle7 clients, 5-21
 - shared structures, 5-21
- Oracle Managed Files, 5-16
- Oracle OLAP, 5-16
- PL/SQL, 5-23

- precompilers, 5-22
- type evolution, 5-34
- UROWIDs, 5-32
- user-defined datatypes, 5-35

J

Java

- downgrading, 7-28, 7-29
- JOB_QUEUE_PROCESSES
 - maximum number of job queue processes, A-5
- JOB_QUEUE_PROCESSES initialization parameter
 - compatibility, A-5
- JVMD817.SQL script, 7-28
- JVMD901.SQL script, 7-28

K

keywords

- behavior differences, 5-26

L

large files

- operating system dependencies, 5-51

large key indexes

- removing, 7-10

LARGE_POOL_SIZE initialization parameter

- changes in behavior, 4-16
- parallel execution allocation, A-8

LEAST_UB function

- desupported, 5-36

listener.ora file

- migrating, B-11
- modifying after upgrading, 4-8
- parameters
 - GLOBAL_DBNAME, B-11
 - ORACLE_HOME, B-11
 - SID_NAME, B-11
- upgrading, B-11

listeners

- configuring for Oracle Enterprise Manager, B-11
- global database name, B-11
- Oracle System Identifier, B-11

- SID, B-11

load balancing

- Net8, 5-50

- LOAD_BALANCE networking parameter, B-7

- LOAD_TNSNAMES command, B-16, B-19

LOB datatype

- copying from LONG, 4-7

LOB index clause

- compatibility, 5-31

LOB retention

- downgrading, 7-7

LOBs

- compatibility, 5-31

locks

- DML lock limit, DML_LOCKS, A-7

LOG_CHECKPOINT_TIMEOUT

- new default value, A-7

LOG_CHECKPOINT_TIMEOUT initialization

- parameter
 - compatibility, A-7

LogMiner

- compatibility, 5-46

LONG datatype

- copying to LOB, 4-7

- LU6.2 protocol, B-3

M

manual upgrade

- advantages, 2-5
- prepare the database, 3-12, D-5

master sites

- incremental upgrade of, E-10
- upgrading, E-3

materialized views

- compatibility, 5-47
- upgrading, 4-16

memory requirements

- for MIG utility, D-3

Messaging Gateway

- downgrading, 7-29

- MGWD901.SQL script, 7-29

MIG utility

- command-line options, D-9
- error messages, D-14

- memory requirements for, D-3
- MULTIPLIER option, D-27
- options
 - CHECK_ONLY, D-9
 - DBNAME, D-9
 - MULTIPLIER, D-9
 - NEW_DBNAME, D-10
 - NO_SPACE_CHECK, D-10
 - PFILE, D-10
 - SPOOL, D-10
- overview, D-2
- running, D-10
 - on UNIX operating systems, D-10
 - on Windows platforms, D-12
- space required for SYSTEM tablespace, D-3
- space requirements for, D-3
- MIGRATE user
 - avoid, D-7
- MIGRATE.BSQ script, D-13
- migration
 - ALTER DATABASE CONVERT statement
 - rerunning, D-30
 - ALTER DATABASE OPEN RESETLOGS
 - MIGRATE statement, 3-22
 - control files, 3-16
 - NCHAR and NLS, 5-33
 - Oracle Managed Files file names, 4-2
 - parallel execution, 4-16
 - ROWID compatibility, D-38
 - rowids, D-32
 - to a different operating system, 3-9
 - troubleshooting
 - datafile version integrity, D-30
- MULTIPLIER
 - MIG utility option, D-9, D-27
- multiversioning, 1-9

N

- NAMES_DID table, B-13
- NAMES_DOM table for Oracle Names, B-13
- NAMES.ADMIN_REGION networking
 - parameter, B-14, B-15
- NAMES.DOMAINS networking parameter, B-13
- namesini.sql script for Oracle Names, B-21

- namesupg.sql script for Oracle Names, B-13, B-15, B-18, B-21
- national character set
 - in Oracle8i, 5-33
- native dynamic SQL
 - interoperability, 5-36
- NCHAR
 - migration, 5-33
 - use in Oracle8i, 5-33
- NCHAR and NLS environment variables
 - compatibility, 5-34
- NCHAR columns
 - upgrading, 4-13
- NCLOBs
 - compatibility, 5-31
- nested tables
 - compatibility, 5-35
- Net8
 - coexistence issues, B-4 to B-6
 - Oracle9i databases, B-4
 - using SERVICE_NAME networking
 - parameter, B-4
 - using SID networking parameter, B-5
 - connection load balancing, 5-50
 - service naming, 5-50
 - SID networking parameter, B-5
 - upgrading to Oracle Names release 1
 - (9.0.1), B-12
 - upgrading to Oracle Net Services, B-8 to B-11
 - configuration files, B-9
 - listener.ora file, B-11
 - software on client, B-9
 - software on server, B-9
 - tnsnames.ora file, B-10
- Net8 OPEN, B-2
- new features
 - adding after upgrade, 4-10
 - requiring a compatibility level, 5-10
- NEW_DBNAME
 - MIG utility option, D-10
- NLS
 - migration, 5-33
- NLS and NCHAR environment variables
 - compatibility, 5-34
- NLS_LANG environment variable

- compatibility, 5-34
- nmcbuild.sql script, B-17
- nmcrgrnt.sql script, B-17
- NMO_INFORMATION table for Oracle Names, B-14
- NO_SPACE_CHECK
 - MIG utility option, D-10
- Novell Directory Services (NDS)
 - authentication, B-2
 - external naming, B-2
- NUMERIC keyword
 - behavior differences, 5-26

O

- O7_DICTIONARY_ACCESSIBILITY initialization parameter
 - compatibility, 5-30, A-7
- object types
 - compatibility, 5-16
 - interoperability, 5-16
- obsolete
 - replication procedures, E-20
- obsolete dynamic performance views, A-22
- obsolete initialization parameters, A-3
- obsolete replication procedures, E-20
- obsolete static data dictionary views, A-16
- OCI
 - applications
 - changing to use Oracle9i, 6-9
 - compatibility, 5-21
 - batch error mode, 5-22
 - client notification, 5-22
 - link line, 5-21
 - LISTEN call and AQ, 5-22
 - thread safety, 5-21
 - interoperability, 5-21
 - Oracle7 clients, 5-21
 - shared structures, 5-21
 - OCIChangePassword call, 5-41
 - OCISessionBegin call, 5-41
 - upgrading applications to Oracle9i, 6-3
- OCI applications
 - upgrading options, 6-6
- OFA, 1-10
- offline datafiles
 - upgrading, D-3
- OLON calls
 - obsolete, 5-21
- operating system
 - migrating to a different, 3-9
- Optimal Flexible Architecture. *See* OFA
- OPTIMAL setting for SYSTEM rollback segment
 - for upgrading, D-26
- optimization
 - compatibility, 5-38
- options
 - deinstalling, 1-12
 - for MIG utility, D-9
- ORA_NLS32 environment variable
 - compatibility, 5-34
- ORA_NLS33 environment variable, D-11
 - compatibility, 5-34
- Oracle Call Interface. *See* OCI
- Oracle Dynamic Services
 - deprecated, 5-17
- Oracle Enterprise Manager
 - static service information in listener.ora file, B-11
- Oracle home
 - multiple, 1-9
- Oracle *interMedia*
 - upgrading, 3-26
- Oracle Managed Files
 - compatibility, 5-16
 - interoperability, 5-16
 - migrating file names, 4-2
- Oracle Media Management API
 - compatibility
 - proxy copy requirement, 5-47
- Oracle Names
 - coexistence issues, B-6
 - migrating
 - ckreg.ora file to cktop.ora file, B-19
 - Oracle Names version 2 using a database, B-13
 - ROSFILES, B-17
 - ROSFILES to a tnsnames.ora file, B-18
 - ROSFILES to Oracle Names tables in database, B-17

- NAMES_DID table for Oracle Names, B-13
- NAMES_DOM table, B-13
- NAMES.ADMIN_REGION parameter in
 - names.ora file, B-14, B-15
- NAMES.DOMAINS parameter in names.ora
 - file, B-13
- namesini.sql script, B-21
- namesupp.sql script, B-13, B-15, B-18, B-21
- NMO_INFORMATION table, B-14
- Oracle Names Control utility commands
 - DUMP_TNSNAMES command, B-15
 - LOAD_TNSNAMES, B-16, B-19
 - REORDER_NS, B-20
- Oracle Names version 2 with Dynamic Discovery
 - Option, B-15
- Oracle Net Manager
 - discovering Oracle Names servers, B-20
 - loading tnsnames.ora file into Oracle
 - Names, B-16, B-19
 - upgrading to release 1 (9.0.1), B-12
- Oracle Names Control utility
 - commands
 - LOAD_TNSNAMES, B-16, B-19
 - REORDER_NS, B-20
- Oracle Net
 - migrating or upgrading to, 5-49
- Oracle Net Services
 - coexistence issues, B-4 to B-6
 - Oracle release 8.0 clients, B-5
 - Oracle release 8.0 databases, B-6
 - third-party applications, B-5
 - using Oracle Net Manager, B-7
 - using SERVICE_NAME networking
 - parameter, B-4
 - using SID networking parameter, B-4
 - FAILOVER networking parameter, B-7
 - listener.ora file with Oracle Enterprise
 - Manager, B-11
 - LOAD_BALANCE networking parameter, B-7
- Oracle Net Manager
 - Use Options Compatible with Net8 8.0 Clients
 - option, B-7
 - Use Oracle8 Release 8.0 Compatible
 - Identification option, B-7
 - SERVICE_NAME parameter, B-4
- SOURCE_ROUTE parameter, B-7
- unsupported features
 - Identix authentication, B-2
 - LU6.2, B-3
 - Net8 OPEN, B-2
 - Novell Directory Services (NDS)
 - authentication, B-2
 - Novell Directory Services (NDS) external
 - naming, B-2
 - prespawnd dedicated servers, B-3
 - protocol.ora file, B-3
 - SecurID authentication, B-2
 - SPX, B-3
 - upgrading to, 4-21
- Oracle OLAP
 - compatibility, 5-16
 - interoperability, 5-16
- Oracle precompilers. *See* precompilers
- Oracle Spatial
 - upgrading, 3-26
- Oracle Syndication Server
 - deprecated, 5-17
- Oracle System Identifier, configuring on the
 - listener, B-11
- Oracle Text
 - migrating from ConText to, D-5
 - upgrading, 3-26
- Oracle Ultra Search
 - upgrading, 3-28
- Oracle Universal Installer, 2-4
- Oracle Visual Information Retrieval
 - upgrading, 3-26
- Oracle Workspace Manager
 - downgrading, 7-29
- ORACLE_HOME initialization parameter, B-11
- ORACLE_TRACE_ENABLE initialization parameter
 - compatibility, A-6
- Oracle9i
 - changes to dynamic performance views, A-20
 - changes to initialization parameters, A-2
 - changes to static data dictionary views, A-14
 - new features
 - adding after upgrade, 4-10
- Oracle9i Real Application Clusters
 - compatibility requirements, 5-39

- downgrading, 7-28
- upgrading, 3-9
- oracle-supplied accounts
 - change passwords, 4-2
- ORADIM
 - downgrading, 7-24
 - upgrading, 3-16
- ORLON calls
 - obsolete, 5-21
- OUTLN user
 - avoid, 3-12
- OWMD901.PLB script, 7-29

P

- PARALLEL clause
 - DEGREE keyword, 5-39
 - INSTANCES keyword removed, 5-39
- parallel execution
 - allocated from large pool, A-8
 - avoiding problems with, 4-16
- parallel table functions
 - removing, 7-20
- parameters
 - for MIG utility. *See* command-line options
- partition views
 - migrate to partition tables, 4-20
- partitioned index-organized tables with LOBs
 - downgrading, 7-12
- password file
 - upgrades
 - exclusive setting, D-28
- password management
 - application changes required for Oracle8i, 5-41
 - interoperability, 5-42
 - password expiration, 5-42
- PDML ITL invariants
 - downgrading, 7-12
- pending transactions
 - and upgrading, D-5
- PFILE
 - MIG utility option, D-10
- pipelined table functions
 - removing, 7-20
- PL/SQL

- backward compatibility, 5-23
- compatibility, 5-23
- functions
 - desupported, 5-36
- integrated SQL analysis, 5-24
- interoperability, 5-23
- PLSQL_V2_COMPATIBILITY initialization
 - parameter, 5-25
- removing incompatibilities for
 - downgrading, 7-20
- variables
 - NCHAR and NLS, 5-33
- precompilers
 - applications
 - changing to use Oracle9i, 6-9
 - upgrading options, 6-6
 - compatibility, 5-22
 - interoperability, 5-22
 - PL/SQL backward compatibility, 5-23
 - upgrading applications to Oracle9i, 6-3
- preparing to upgrade, 2-2
- prespawnd dedicated servers, B-3
- primary keys
 - upgrading snapshots, E-15
- Pro*Ada
 - upgrading to SQL*Module for Ada, 5-23
- Pro*C/C++
 - connecting with SYSDBA privileges, 5-22
- Pro*COBOL
 - connecting with SYSDBA privileges, 5-23
- Procedural Option
 - required for upgrading, D-5
- PROPSS view
 - NCHAR and NLS, 5-33
- protocol.ora file, B-3
- proxy copy
 - requirement, 5-47

Q

- queue tables
 - upgrading, 4-17

R

RAW

- maximum size
- change, 5-19

read-only tablespaces

- upgrading, D-3

REAL keyword

- behavior differences, 5-26

recovery

- compatibility, 5-43

recovery catalog

- compatibility with Recovery Manager, 5-43
- upgrading, 4-18

Recovery Manager

- backup management, 5-44
- commands
- compatibility, 5-43
- compatibility, 5-43
- normalize catalog, 4-11

release

- definition, 1-2

release 9.0 partitioning methods

- downgrading, 7-10

release 9.2 DEFAULT partitions

- downgrading, 7-4

release 9.2 partitioning methods

- downgrading, 7-5

release 9.2 Streams

- downgrading, 7-5

release 9.2 subpartition templates

- downgrading, 7-6

releases

- multiple, 1-9

relinking with SQL*Net, 6-3

REMOTE_LOGIN_PASSWORDFILE initialization

- parameter, 3-19
- upgrading, D-28

renamed initialization parameters, D-39

REORDER_NS command, B-20

replication

- compatibility, 5-48
- database migration overview, E-2
- downgrading, 7-28
- obsolete procedures, E-20

upgrading, 3-11, D-5

requirements

- Export utility, 8-3
- import utility, 8-3

restricted ROWIDs, D-38

ROLLBACK FORCE command, D-5

rollback segments

- upgrading, D-3

rosbild.sql script, B-17

ROSFILES

- nmcbuild.sql script, B-17
- nmcgrnt.sql script, B-17
- rosbild.sql script, B-17
- rosgnt.sql script, B-17

rosgnt.sql script, B-17

ROWIDs

- extended, D-38
- restricted, D-38

rowids

- compatibility, 5-32
- client access, D-37
- conversion from Oracle7 format, D-34
- examples, D-36
- DBMS_ROWID compatibility package, D-34
- migration, D-32
- snapshot refresh, D-37

S

SAVEPOINT keyword

- behavior differences, 5-26

schema objects

- compatibility, 5-30

scripts

- downgrading, 7-22
- rerunning, 7-22
- upgrading, 3-22

SecurID authentication, B-2

SERIALIZABLE initialization parameter

- compatibility, A-6

Server Manager

- differences with SQL*Plus
- ampersands, C-12
- blank lines, C-9
- commands, C-3

- comments, C-7
 - COMMIT command, C-14
 - CREATE LIBRARY command, C-13
 - CREATE TYPE command, C-13
 - hyphen continuation character, C-10
 - startup, C-2
 - syntax, C-7
- migrating scripts to SQL*Plus, C-1
- not supported in Oracle9i, 4-15, C-1
- server parameter file
 - migrating to, 4-15
- service naming
 - Net8, 5-50
- SERVICE_NAME parameter, B-4
- SERVICE_NAMES initialization parameter, B-8
- SET COMPATIBILITY command
 - SQL*Plus scripts, 6-10
- SET_SESSION_LONGOPS procedure
 - syntax change, 5-38
- shared server
 - requirements for running, 5-49
- shared structures
 - interoperability, 5-21
- SHARED_POOL_SIZE initialization parameter
 - changes in behavior, 4-16
 - parallel execution allocation, A-8
- SID networking parameter, B-5
- SID, configuring on the listener, B-11
- SID_NAME parameter, B-11
- snapshot sites
 - upgrading, E-8
- snapshots
 - refresh
 - physical ROWIDs, D-37
 - upgrading to primary key, E-15
- SORT_AREA_SIZE initialization parameter
 - compatibility, A-6
- SORT_DIRECT_WRITES initialization parameter
 - compatibility, A-6
- SOURCE_ROUTE parameter, B-7
- space requirements
 - for MIG utility, D-3
- SPOOL
 - MIG utility option, D-10
- SPX protocol, B-3
- SQL
 - removing incompatibilities for
 - downgrading, 7-20
- SQL commands
 - COMMIT FORCE, D-5
 - ROLLBACK FORCE, D-5
- SQL*Module
 - for Ada, 5-23
- SQL*Net
 - coexistence issues, B-4 to B-6
 - Oracle9i databases, B-4
 - using SERVICE_NAME networking
 - parameter, B-4
 - using SID networking parameter, B-5
 - migrating to Oracle Net, 5-49
 - migrating to Oracle Net Services, B-8 to B-11
 - configuration files, B-9
 - listener.ora file, B-11
 - software on client, B-9
 - software on server, B-9
 - tnsnames.ora file, B-10
 - verifying service name and instance
 - name, B-8
 - relinking, 6-3
 - SID networking parameter, B-5
 - upgrading from V1 to V2, 5-49
 - upgrading to Oracle Names release 1
 - (9.0.1), B-12
 - upgrading to Oracle Net Services, 4-21
 - use with Oracle9i, 6-3
- SQL*Plus
 - differences with Server Manager
 - ampersands, C-12
 - blank lines, C-9
 - commands, C-3
 - comments, C-7
 - COMMIT command, C-14
 - CREATE LIBRARY command, C-13
 - CREATE TYPE command, C-13
 - hyphen continuation character, C-10
 - startup, C-2
 - syntax, C-7
 - migrating scripts from Server Manager, C-1
 - scripts
 - upgrading, 6-9

- standby database
 - compatibility, 5-45
 - upgrading, 4-8
- STARTUP
 - compatibility, 5-27
- statements
 - ALTER DATABASE RESET COMPATIBILITY, 5-10
 - ALTER DATABASE RESET LOGS, D-5
- static data dictionary views
 - changes, D-40
 - changes in Oracle9i, A-14
 - deprecated, A-14
 - obsolete, 5-30, A-16
 - with columns that may return nulls, A-18
 - with dropped columns, A-17
 - with renamed columns, A-16
- statistics tables
 - upgrading, 4-19
- SUBSTR operator, 5-19
- subtypes
 - downgrading, 7-19
- SYS schema
 - user-created objects in, 5-30
- SYSDBA
 - connecting in Pro*C/C++, 5-22
 - connecting in Pro*COBOL, 5-23
- SYSTEM tablespace
 - MIG utility, D-3
 - space
 - insufficient for upgrading, D-25

T

- tablespaces
 - automatic segment-space managed
 - removing, 7-8
 - compatibility, 5-27
 - upgrading offline tablespaces, D-6
- tempfiles
 - data dictionary information, 5-29
- temporary tablespace
 - space
 - insufficient for upgrading, D-25
- terminology
 - database migration, 1-2
- testing
 - applications for upgrade, 2-12
 - developing a plan, 2-8
 - EXPLAIN PLAN, 2-11
 - functional for upgrade, 2-9
 - integration for upgrading, 2-9
 - INTO clause, 2-11
 - minimal for upgrade, 2-9
 - performance for upgrade, 2-10
 - pre-upgrade and post-upgrade, 2-11
 - the upgrade process, 2-12
 - the upgraded test database, 2-12
 - upgrading results, 4-22
 - volume/load stress for upgrade, 2-10
- thread safety
 - compatibility, 5-21
- tnsnames.ora file
 - migrating, B-10
 - parameters
 - FAILOVER, B-7
 - LOAD_BALANCE, B-7
 - SERVICE_NAME, B-4
 - SID, B-5
 - SOURCE_ROUTE, B-7
 - upgrading, B-10
- TO_LABEL function
 - desupported, 5-36
- TO_LOB function, 4-7
- transactions
 - pending, D-5
- Transparent Application Failover (TAF)
 - GLOBAL_DBNAME networking parameter in listener.ora, B-11
- troubleshooting
 - database fails to open after removing incompatibilities, 5-10
 - migration
 - datafile version integrity, D-30
 - upgrades
 - ALTER DATABASE CONVERT statement, D-27
 - missing convert file, D-28
 - MULTIPLIER option, D-27
 - NOMOUNT database start mode, D-28

- Oracle7 control file, D-27
- upgrading
 - AUDIT_TRAIL initialization parameter, D-25
 - database name mismatch, D-29
 - MIG utility error messages, D-14
 - OPTIMAL setting, D-26
 - password file, D-28
 - running the MIG utility, D-25
 - SYSTEM tablespace, D-25
 - temporary tablespace, D-25
- TRUE keyword
 - behavior differences, 5-26
- tuning
 - after upgrading, 4-22
- type evolution
 - interoperability, 5-34
- types
 - evolved
 - removing, 7-19
 - non-final
 - downgrading, 7-19

U

- undo tablespaces
 - removing, 7-9
- upgrade
 - troubleshooting
 - MIG utility error messages, D-14
- upgrade methods
 - choosing, 2-3
 - copying data, 2-7
 - Database Upgrade Assistant, 2-4
 - Export/Import, 2-6
 - manual upgrade, 2-5
- upgrading
 - abandoning, D-31
 - Advanced Queuing, 4-17
 - after upgrading, 4-1
 - ALTER DATABASE CONVERT statement, 3-21
 - applications, 6-1
 - compatibility rules, 6-4
 - options, 6-6
 - relinking, 6-4

- AUDIT_TRAIL initialization parameter, D-25
- backup strategy, 2-8
- character set, 3-10
- control files, 3-15
- exclusive password file, D-28
- features requiring, E-18
- incremental, E-6
- initialization parameters, 3-18
- listener.ora file, 4-8
- master sites, E-3
- materialized views, 4-16
- MIGRATE user, avoid, D-7
- MIGRATE.BSQ script, D-13
- NCHAR columns, 4-13
- new administrative procedures, 4-10
- offline datafiles, D-3
- offline tablespaces, D-6
- OPTIMAL setting for SYSTEM rollback segment, D-26
- Oracle Forms applications, 6-10
- Oracle *interMedia*, 3-26
- Oracle Spatial, 3-26
- Oracle Text, 3-26
- Oracle Ultra Search, 3-28
- Oracle Visual Information Retrieval, 3-26
- Oracle9i Real Application Clusters, 3-9
- ORADIM, 3-16
- OUTLN user, avoid, 3-12
- parallel execution, 4-16
- post upgrade actions, 4-1
- queue tables, 4-17
- read-only tablespaces, D-3
- recovery catalog, 4-18
- replication, 3-11, D-5
- rolling upgrades, 1-12
- scripts, 3-22
 - CATALOG.SQL, 3-23, D-3
 - CATPROC.SQL, 3-23, D-3
- snapshot sites, E-8
- specific components, 3-25
- SQL*Plus scripts, 6-9
- standby database, 4-8
- statistics tables, 4-19
- SYSTEM tablespace, D-25
- temporary tablespace, D-25

- testing, 2-8
- testing results, 4-22
- to primary key snapshots, E-15
- troubleshooting
 - ALTER DATABASE CONVERT statement, D-27
 - AUDIT_TRAIL initialization parameter, D-25
 - database name mismatch, D-29
 - missing convert file, D-28
 - MULTIPLIER option, D-27
 - NOMOUNT database start mode, D-28
 - OPTIMAL setting, D-26
 - Oracle7 control file, D-27
 - password file, D-28
 - running the MIG utility, D-25
 - SYSTEM tablespace, D-25
 - temporary tablespace, D-25
- tuning after, 4-22
- using the Database Upgrade Assistant, 3-4
- upgrading a database
 - ALTER DATABASE CONVERT statement, D-2
 - ALTER DATABASE OPEN RESETLOGS MIGRATE statement, D-3
 - block size minimums, D-4
 - choosing an upgrade method, 2-3
 - distributed database considerations, D-5
 - manually, 3-9
 - prepare the database, 3-12, D-5
 - overview of steps, 1-4
 - overview of the MIG utility, D-2
 - performing a manual upgrade, 2-4
 - preparing to, 2-2
 - role of application developer, 1-8
 - role of database administrator, 1-7
 - rollback segments, D-3
 - using Export/Import, 8-3
 - using the Database Upgrade Assistant, 2-3
- UROWIDs
 - interoperability, 5-32
- Use Options Compatible with Net8 8.0 Clients option, B-7
- Use Oracle8 Release 8.0 Compatible Identification option, B-7
- user-created objects

- in SYS schema, 5-30
- user-defined aggregate functions
 - dropping for downgrading, 7-19
- user-defined datatypes
 - compatibility, 5-34
 - interoperability, 5-35
 - new format, 5-35
- UTLCHN1.SQL script, 5-36
- UTLCONST.SQL script, 4-20
- UTLEXPT1.SQL script, 5-37
- UTLINCMP.SQL script, 7-3

V

- VALUES view
 - NCHAR and NLS, 5-33
- VARCHAR2
 - maximum size
 - change, 5-19
- varrays
 - stored as LOBs
 - compatibility, 5-35

W

- word size
 - changing, 1-11, 4-22

X

- XMLD817.SQL script, 7-29
- XMLD901.SQL script, 7-29