Oracle® Database

Database Upgrade Guide

19c

E96252-36

May 2021

Oracle Database Database Upgrade Guide, 19c

E96252-36

Copyright © 1996, 2021, Oracle and/or its affiliates.

Primary Author: Douglas Williams

Contributors: Drew Adams, Pablo Sainz Albanez, Frederick Alvarez, Yasin Baskan, Subhransu Basu, Rae Burns, Rhonda Day, Mike Dietrich, Joseph Errede, Daniel Overby Hansen, Yuan Hao, Subrahmanyam Kodavaluru, Jai Krisnani, Cindy Lim, Valarie Moore, Byron Motta, Satish Panchumarthy, Geetha Ravi, Carol Tagliaferri, Hector Vieyra

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government’s use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience xxv
Documentation Accessibility xxv
Set Up Java Access Bridge to Implement Java Accessibility xxvi
Related Documentation xxvi
Conventions xxvi

1 Introduction to Upgrading Oracle Database

Overview of Oracle Database Upgrade Tools and Processes 1-1
- Definition of Terms Upgrading and Migrating 1-1
- Upgrade and Data Migration Methods and Processes 1-2
- Oracle Database Releases That Support Direct Upgrade 1-3
- Where to Find the Latest Information About Upgrading Oracle Database 1-5

Major Steps in the Upgrade Process for Oracle Database 1-6
Compatibility and Interoperability Between Oracle Database Releases 1-10
- About Oracle Database Release Numbers 1-11
- Convention for Referring to Release Numbers in Oracle Database Upgrade Guide 1-13
- What Is Oracle Database Compatibility? 1-13
  - Understanding Oracle Database Compatibility 1-13
  - When to Set the COMPATIBLE Initialization Parameter in Oracle Database 1-14
  - The COMPATIBLE Initialization Parameter in Oracle Database 1-14
  - Values for the COMPATIBLE Initialization Parameter in Oracle Database 1-16
  - About Downgrading and Compatibility for Upgrading Oracle Database 1-16
  - How the COMPATIBLE Initialization Parameter Operates in Oracle Database 1-16
  - Checking the Compatibility Level of Oracle Database 1-17
- What Is Interoperability for Oracle Database Upgrades? 1-17
- About Invalid Schema Objects and Database Upgrades 1-18
- About Upgrading Oracle OLAP Data Security Policies 1-19

About Running Multiple Oracle Releases 1-19
- Databases in Multiple Oracle Homes on Separate Computers 1-20

Databases in Multiple Oracle Homes on the Same Computer 1-20
About the Optimal Flexible Architecture Standard 1-21
About Multiple Oracle Homes Support 1-21
About Converting Databases During Upgrades 1-22
Overview of Converting Databases During Upgrades 1-23
About Upgrading Using Standby Databases 1-25
Migrating From Standard Edition to Enterprise Edition of Oracle Database 1-25
Migrating from Enterprise Edition to Standard Edition of Oracle Database 1-27
Migrating from Oracle Database Express Edition (Oracle Database XE) to Oracle Database 1-27
About Upgrading Platforms for a New Oracle Database Release 1-28
About Upgrading Your Operating System 1-28
Options for Transporting Data to a Different Operating System 1-28
About Image-Based Oracle Database Installation 1-29

2 Preparing to Upgrade Oracle Database

Tasks to Prepare for Oracle Database Upgrades 2-2
Review Deprecated and Desupported Features 2-2
Become Familiar with New Oracle Database Features 2-3
Choose an Upgrade Method for Oracle Database 2-3
The AutoUpgrade Utility Method for Upgrading Oracle Database 2-4
The Graphical User Interface Method for Upgrading Oracle Database 2-4
The Manual, Command-Line Method for Upgrading Oracle Database 2-4
The Export/Import Method for Migrating Data When Upgrading Oracle Database 2-5
Choose a New Location for Oracle Home when Upgrading 2-6
Develop a Test Plan for Upgrading Oracle Database 2-6
Upgrade Testing 2-7
Minimal Testing 2-7
Functional Testing After Upgrades 2-8
High Availability Testing 2-8
Integration Testing to Ensure Applications are Compatible 2-8
Performance Testing an Upgraded Oracle Database 2-9
Volume and Load Stress Testing for Oracle Database Upgrades 2-12
Test Plan Guidelines for Oracle Database Upgrade Planning 2-13
Schema-Only Accounts and Upgrading EXPIRED Password Accounts 2-13
Back Up Files to Preserve Downgrade and Recovery Options 2-14
Prepare a Backup Strategy Before Upgrading Oracle Database 2-14
Oracle Data Guard Broker Configuration File and Downgrades 2-15
Exporting a Broker Configuration 2-15
Pre-Upgrade Information Tool Checklist 2-16

Installing the New Oracle Database Software for Single Instance 2-16
Installing the New Oracle Database Software for Oracle RAC 2-17
Database Preparation Tasks to Complete Before Starting Oracle Database Upgrades 2-18
Release Updates and Requirements for Upgrading Oracle Database 2-19
Copying Transparent Encryption Oracle Wallets 2-19
Recommendations for Oracle Net Services When Upgrading Oracle Database 2-20
Understanding Password Case Sensitivity and Upgrades 2-21
Checking for Accounts Using Case-Insensitive Password Version 2-22
Check for Profile Scripts (glogin.sql and login.sql) 2-25
Running Upgrades with Read-Only Tablespaces 2-26
High Availability Options for Oracle Database 2-27
Options for High Availability with Oracle Database Standard Edition 2-28
Preparing to Upgrade Standard Edition Oracle RAC or Oracle RAC One Node 2-28
Requirements for Using Standard Edition High Availability With Oracle Databases 2-28
Non-CDB Upgrades and Oracle GoldenGate 2-29
Preparing for Database Rolling Upgrades Using Oracle Data Guard 2-29
Preparing the New Oracle Home for Upgrading 2-30
Prerequisites for Preparing Oracle Home on Windows 2-32
Performing Preupgrade Checks Using AutoUpgrade 2-33
About AutoUpgrade Utility System Checks 2-33
Example of Running AutoUpgrade Prechecks Using Analyze Mode 2-34
Checking the Upgrade Checks Overview File 2-36
Creating a Configuration File to Run AutoUpgrade Prechecks On a CDB 2-37
Running AutoUpgrade Fixups on the Earlier Release Oracle Database 2-38
Testing the Upgrade Process for Oracle Database 2-38
Example of Testing Upgrades Using Priority List Emulation 2-39
Upgrade Oracle Call Interface (OCI) and Precompiler Applications 2-41
Requirements for Upgrading Databases That Use Oracle Label Security and Oracle Database Vault 2-41
Audit Table Preupgrade and Archive Requirements 2-42
DBUA, AutoUpgrade, and Oracle Database Vault 2-43
Oracle Database Vault and Upgrades of Oracle Database Release 11.2 2-43
Granting the DV_PATCH_ADMIN Role to SYS for Oracle Database Vault 2-44
Backing Up Oracle Database for Upgrading 2-44

3 Using AutoUpgrade for Oracle Database Upgrades

About Oracle Database AutoUpgrade 3-2
Checking for Java Version Support 3-3

Examples of How to Use AutoUpgrade 3-4
AutoUpgrade with Source and Target Database Homes on Same Server (Typical) 3-5
AutoUpgrade with Source and Target Database Homes on Different Servers 3-5
AutoUpgrade Messages and Process Description Terms 3-6
Overview of AutoUpgrade Job IDs 3-6
Overview of AutoUpgrade Stages 3-6
Overview of AutoUpgrade Stage Operations and States 3-7
About AutoUpgrade Processing Modes 3-7
Preparations for Running AutoUpgrade Processing Modes 3-8
About the AutoUpgrade Analyze Processing Mode 3-9
About the AutoUpgrade Fixups Processing Mode 3-10
About the AutoUpgrade Deploy Processing Mode 3-11
About the AutoUpgrade Upgrade Processing Mode 3-12
Understanding AutoUpgrade Workflows and Stages 3-12
Understanding Non-CDB to PDB Upgrades with AutoUpgrade 3-14
AutoUpgrade Command-Line Parameters 3-15
AutoUpgrade Command-Line Syntax 3-16
Debug 3-19
Clear_recovery_data 3-19
Config 3-20
Config_Values 3-21
Console 3-24
Create_sample_file 3-26
Error_code 3-28
Mode 3-29
Noconsole 3-30
Preupgrade 3-31
Settings 3-34
Version 3-34
Restore 3-35
Restore_on_fail 3-35
Zip 3-36
AutoUpgrade Utility Configuration Files 3-36
Global Parameters for the AutoUpgrade User Configuration File 3-37
Local Parameters for the AutoUpgrade Configuration File 3-43
Locally Modifiable Global Parameters for AutoUpgrade Configuration File 3-56
AutoUpgrade and Oracle Database Configuration Options 3-60
Non-CDB to PDB Upgrade Guidelines and Examples 3-60
AutoUpgrade Process Flow for Oracle Grid Infrastructure Configurations 3-61
Oracle RAC Requirements for Upgrade with AutoUpgrade 3-62

AutoUpgrade and Oracle Data Guard 3-63
How AutoUpgrade Performs Oracle Data Guard Upgrades 3-63
Steps AutoUpgrade Completes for Oracle Data Guard Upgrades 3-64
Steps After the Primary Database is Upgraded 3-64
AutoUpgrade Configuration File Examples 3-65
Updating the TDE Wallet Store Location During Upgrade Using AutoUpgrade 3-66
AutoUpgrade Configuration File with Two Database Entries 3-67
Standardizing Upgrades With AutoUpgrade Configuration File Entries 3-68
AutoUpgrade Configuration File for Incremental Upgrade of a Set of PDBs 3-70
How to Run AutoUpgrade in a Script or Batch job 3-71
AutoUpgrade Internal Settings Configuration File 3-71
AutoUpgrade Log File Structure 3-73
Enabling Full Deployments for AutoUpgrade 3-76
Examples of How to Use the AutoUpgrade Console 3-77
How to Override Default Fixups 3-78
Local Configuration File Parameter Fixups Checklist Example 3-83
Proper Management of AutoUpgrade Database Changes 3-83
AutoUpgrade and Microsoft Windows ACLs and CLIs 3-84

4 Upgrading Oracle Database Manually Using Parallel Upgrade Utility

Rerunning Upgrades for Oracle Database 4-1
About Rerunning Upgrades for Oracle Database 4-2
Rerunning Upgrades with the Upgrade (catctl.pl) Script 4-2
Options for Rerunning the Upgrade for Multitenant Databases (CDBs) 4-5
Rerun the Entire Upgrade for the CDB 4-6
Rerun the Upgrade Only on Specified PDBs 4-6
Rerun the Upgrade While Other PDBs Are Online 4-8
Rerun the Upgrade Using an Inclusion List to Specify a CDB or PDBs 4-10
Restarting the Upgrade from a Specific Phase that Failed Using -p 4-11
Reviewing CDB Log Files for Failed Phases 4-11
Procedure for Finding and Restarting Multitenant Upgrades from a Failed Phase 4-12
Upgrading Manually with Parallel Upgrade Utility 4-14
About the Parallel Upgrade Utility for Oracle Database (CATCTL.PL and DBUPGRADE) 4-14
General Steps for Running the Parallel Upgrade Utility 4-15
Parallel Upgrade Utility (catctl.pl) Parameters 4-17
Example of Using the Parallel Upgrade Utility 4-20
Manual Upgrade Scenarios for Multitenant Architecture Oracle Databases 4-22
About Oracle Multitenant Oracle Database Upgrades 4-23
Coordinate Upgrades of Proxy PDBs with Multitenant Upgrades 4-23

Manually Upgrading a Multitenant Container Oracle Database (CDB) 4-24
About Upgrading PDBs Using the Parallel Upgrade Utility with Priority Lists 4-30
About PDB Upgrades Using Priority Lists, Inclusion Lists, and Exclusion Lists 4-31
Upgrading Multitenant Architecture In Parallel 4-37
About Upgrading Pluggable Databases (PDBs) In Parallel 4-37
Upgrading Multitenant Container Databases In Parallel 4-39
Upgrading Multitenant Architecture Sequentially Using Unplug-Plug 4-44
About Upgrading Pluggable Databases (PDBs) Sequentially 4-45
Unplugging the Earlier Release PDB from the Earlier Release CDB 4-46
Plugging in the Earlier Release PDB to the Later Release CDB 4-47
Upgrading the Earlier Release PDB to the Later Release 4-48
Use Inclusion or Exclusion Lists for PDB Upgrades 4-49
Example of Manual Upgrade of Windows Non-CDB Oracle Database 4-49
Preparing to Upgrade Windows Non-CDB Using Command-Line Utilities 4-50
Manually Upgrading Windows Non-CDB Using Command-Line Utilities 4-57
Running Postupgrade Fixup Scripts After Upgrading a Non-CDB Database 4-65
About Dbupgrade Scripts and catupgrd.sql in Earlier Releases of Oracle Database 4-72
About Transporting and Upgrading a Database (Full Transportable Export/Import) 4-73
Upgrade Scenarios for Non-CDB Oracle Databases 4-73
Download the Latest Oracle Database Preupgrade Information Tool 4-74
About Adopting a Non-CDB as a PDB Using a PDB Plugin 4-75
Adopting a Non-CDB as a PDB 4-76
Oracle Label Security Integration in a Multitenant Environment 4-79
Manually Upgrading Non-CDB Architecture Oracle Databases 4-79
Upgrading Oracle Database Using Fleet Patching and Provisioning 4-85

5 Troubleshooting the Upgrade for Oracle Database

About Starting Oracle Database in Upgrade Mode 5-2
Running DBUA with Different ORACLE_HOME Owner 5-3
Invalid Object Warnings and DBA Registry Errors 5-3
Invalid Objects and Premature Use of Postupgrade Tool 5-4
Resolving Oracle Database Upgrade Script Termination Errors 5-4
Troubleshooting Causes of Resource Limits Errors while Upgrading Oracle Database 5-4
Resolving SQL*Plus Edition Session Startup Error for Oracle Database 5-6
Error ORA-00020 Maximum Number of Processes Exceeded When Running utlrp.sql 5-6
Fixing ORA-28365: Wallet Is Not Open Error 5-7
Resolving issues with view CDB_JAVA_POLICY 5-7
Continuing Upgrades After Server Restarts (ADVM/ACFS Driver Error) 5-8
Component Status and Upgrades 5-8

Understanding Component Status With the Post-Upgrade Status Tool 5-9
Component OPTION OFF Status and Upgrades 5-10
Example of an Upgrade Summary Report 5-11
Standard Edition Starter Database and Components with Status OPTION OFF 5-12
Adjusting Oracle ASM Password File Location After Upgrade 5-12
Fixing "Warning XDB Now Invalid" Errors with Pluggable Database Upgrades 5-13
Fixing ORA-27248: sys.dra_reevaluate_open_failures is running 5-14
Fixing Failed Upgrades Where Only Datapatch Fails 5-14

6 Post-Upgrade Tasks for Oracle Database

Release Updates and Requirements for Upgrading Oracle Database 6-1
Check the Upgrade With Post-Upgrade Status Tool 6-2
How to Show the Current State of the Oracle Data Dictionary 6-2
Required Tasks to Complete After Upgrading Oracle Database 6-3
Setting Environment Variables on Linux and Unix Systems After Manual Upgrades 6-5
Recompiling All Invalid Objects 6-5
Recompiling All Invalid Objects on Multitenant Architecture Databases 6-6
Track Invalid Object Recompilation Progress 6-6
Setting oratab and Scripts to Point to the New Oracle Location After Upgrading Oracle Database 6-7
Check PL/SQL Packages and Dependent Procedures 6-8
Upgrading Tables Dependent on Oracle-Maintained Types 6-8
Adjusting Minimum and Maximum for Parallel Execution Servers 6-9
About Recovery Catalog Upgrade After Upgrading Oracle Database 6-9
Upgrading Statistics Tables Created by the DBMS_STATS Package After Upgrading Oracle Database 6-9
Upgrading Externally Authenticated SSL Users After Upgrading Oracle Database 6-10
Configuring the FTP and HTTP Ports and HTTP Authentication for Oracle XML DB 6-11
Install Oracle Text Supplied Knowledge Bases After Upgrading Oracle Database 6-12
Rebuild Oracle Text Indexes Using AUTO_LEXER 6-12
Update Oracle Application Express Configuration After Upgrading Oracle Database 6-13
Replace the DEMO Directory in Read-Only Oracle Homes 6-14
Configure Access Control Lists (ACLs) to External Network Services 6-15
Enabling Oracle Database Vault After Upgrading Oracle Database 6-15
Upgrading Oracle Database Without Disabling Oracle Database Vault 6-15
Common Upgrade Scenarios with Oracle Database Vault 6-16
Check for the SQLNET.ALLOWED_LOGON_VERSION Parameter Behavior 6-17
Recommended and Best Practices to Complete After Upgrading Oracle Database 6-17

Back Up the Database 6-19
Running Postupgrade Fixup Scripts 6-19
Gathering Dictionary Statistics After Upgrading 6-21
Regathering Fixed Objects Statistics with DBMS_STATS 6-22
Reset Passwords to Enforce Case-Sensitivity 6-22
Finding and Resetting User Passwords That Use the 10G Password Version 6-23
Understand Oracle Grid Infrastructure, Oracle ASM, and Oracle Clusterware 6-25
Oracle Grid Infrastructure Installation and Upgrade and Oracle ASM 6-26
Add New Features as Appropriate 6-26
Develop New Administrative Procedures as Needed 6-27
Migrating From Rollback Segments To Automatic Undo Mode 6-27
Migrating Tables from the LONG Data Type to the LOB Data Type 6-28
Migrate Your Upgraded Oracle Databases to Use Unified Auditing 6-28
Understanding Unified Auditing Migration Process for Oracle Database 6-29
Migrating to Unified Auditing for Oracle Database 6-30
About Managing Earlier Audit Records After You Migrate to Unified Auditing 6-32
Removing the Unified Auditing Functionality 6-32
Obtaining Documentation References if You Choose Not to Use Unified Auditing 6-33
Identify Oracle Text Indexes for Rebuilds 6-33
Dropping and Recreating DBMS_SCHEDULER Jobs 6-34
Transfer Unified Audit Records After the Upgrade 6-34
About Transferring Unified Audit Records After an Upgrade 6-34
Transferring Unified Audit Records After an Upgrade 6-35
About Testing the Upgraded Production Oracle Database 6-36
Upgrading the Time Zone File Version After Upgrading Oracle Database 6-37
Recommended Tasks After Upgrading an Oracle RAC Database 6-37
Recommended Tasks After Upgrading Oracle ASM 6-37
Create a Shared Password File In the ASM Diskgroup 6-38
Reset Oracle ASM Passwords to Enforce Case-Sensitivity 6-38
Advancing the Oracle ASM and Oracle Database Disk Group Compatibility 6-39
Set Up Oracle ASM Preferred Read Failure Groups 6-39
Recommended Tasks After Upgrading Oracle Database Express Edition 6-40
Tasks to Complete Only After Manually Upgrading Oracle Database 6-40
Changing Passwords for Oracle Supplied Accounts 6-41
Create or Migrate Your Password File with ORAPWD 6-41
Migrating Your Initialization Parameter File to a Server Parameter File 6-42
Identifying and Copying Oracle Text Files to a New Oracle Home 6-42
Upgrading the Oracle Clusterware Configuration 6-43
Adjust the Initialization Parameter File for the New Release 6-43
Setting the COMPATIBLE Initialization Parameter After Upgrade 6-44

Adjust TNSNAMES.ORA and LISTENER Parameters After Upgrade 6-45
Set CLUSTER_DATABASE Initialization Parameter For Oracle RAC After Upgrade 6-45

7 Upgrading Applications After Upgrading Oracle Database

Schema-Only Accounts and Upgrading EXPIRED Password Accounts 7-2
Overview of Upgrading Applications on a New Oracle Database Release 7-2
Compatibility Issues for Applications on Different Releases of Oracle Database 7-3
Software Upgrades and Client and Server Configurations for Oracle Database 7-3
Possible Client and Server Configurations for Oracle Database 7-3
Types of Software Upgrades for Oracle Database Client and Server Software 7-4
Compatibility Rules for Applications When Upgrading Oracle Database Client or Server Software 7-4
Rules for Upgrading Oracle Database Server Software 7-5
If You Do Not Change the Client Environment, Then You Are Not Required to Relink 7-5
Applications Can Run Against Newer or Older Oracle Database Server Releases 7-6
Upgrading the Oracle Database Client Software 7-6
About Image-Based Oracle Database Client Installation 7-7
About Linking Applications with Newer Libraries 7-7
Statically Linked Applications Must Always Be Relinked 7-7
About Relinking Dynamically Linked Applications 7-7
About Upgrading Precompiler and OCI Applications in Oracle Database 7-8
About Upgrading Options for Oracle Precompiler and OCI Applications 7-8
Option 1: Leave the Application Unchanged 7-9
Option 2: Precompile or Compile the Application Using the New Software 7-9
Option 3: Change the Application Code to Use New Oracle Database Features 7-10
Changing Oracle Precompiler and OCI Application Development Environments 7-10
Changing Precompiler Applications 7-10
Changing OCI Applications 7-11
Upgrading SQL*Plus Scripts and PL/SQL after Upgrading Oracle Database 7-11
About Upgrading Oracle Forms or Oracle Developer Applications 7-11

8 Downgrading Oracle Database to an Earlier Release

Supported Releases for Downgrading Oracle Database 8-2
Check for Incompatibilities When Downgrading Oracle Database 8-4
Perform a Full Backup Before Downgrading Oracle Database 8-4
Performing Required Predowngrade Steps for Oracle Database 8-5
Using Scripts to Downgrade a CDB or Non-CDB Oracle Database 8-7

Using Dbdowngrade to Downgrade CDB and Non-CDB Oracle Databases 8-7
Downgrading a CDB or Non-CDB Oracle Database Manually with catdwgrd.sql 8-10
About Downgrades and Invalid Objects with Component Status OPTION OFF 8-18
Downgrading a Single Pluggable Oracle Database (PDB) 8-18
Downgrading PDBs That Contain Oracle Application Express 8-20
Post-Downgrade Tasks for Oracle Database Downgrades 8-20
Oracle XML DB Authentication Recommendations for an Oracle Database Downgrade 8-21
Reapply Release Update and Other Patches After Downgrade 8-21
Re-enabling Oracle Database Vault after Downgrading Oracle Database 8-22
Restoring the Configuration for Oracle Clusterware 8-22
Restoring Oracle Enterprise Manager after Downgrading Oracle Database 8-22
Requirements for Restoring Oracle Enterprise Manager After Downgrading 8-23
Running EMCA to Restore Oracle Enterprise Manager After Downgrading 8-23
Running the emdwgrd utility to restore Enterprise Manager Database Control 8-26
Restoring Oracle Application Express to the Earlier Release 8-27
Gathering Dictionary Statistics After Downgrading 8-27
Regathering Fixed Object Statistics After Downgrading 8-28
Regathering Stale CBO Statistics After Downgrade 8-29
Troubleshooting the Downgrade of Oracle Database 8-29
Errors Downgrading Oracle Database Components with catdwgrd.sql Script 8-30
Oracle Multimedia Downgrade and imrelod.sql Script Error 8-33
Oracle Database Vault and dvrelod.sql Script Error 8-33
Downgrading Oracle Grid Infrastructure (Oracle Restart) After Successful or Failed Upgrade 8-33
Oracle ACFS and Oracle Grid Infrastructure Downgrades to 11g Release 2 (11.2) 8-33
Database Links Passwords After Downgrading Oracle Database 11g Release 1 (11.1) 8-34
ORA-00600 Errors with Database Links Passwords After Downgrading to Oracle Database 11.1 Release 1 8-34
Using Oracle Data Pump Export to Create a Dump File Containing All Existing Database Links 8-34

9 Migrating Data Using Oracle Data Pump

Overview of Data Pump and Export/Import For Migrating Data 9-1
Migrating Data With Oracle Data Pump Before Upgrades 9-2
Importing a Full Oracle Database Using a Network Link 9-4
Data Pump Requirements When Downgrading Oracle Database 9-5

10 Behavior Changes, Deprecated and Desupported Features for Oracle Database

About Deprecated and Desupported Status 10-1
Behavior Changes, Deprecations and Desupports in Oracle Database 19c 10-1
- Behavior Changes for Oracle Database 19c Upgrade Planning 10-2
  - Changes to Oracle Data Guard Properties Management 10-2
  - Rapid Home Provisioning (RHP) Name Change 10-3
  - Resupport of Direct File Placement for OCR and Voting Disks 10-3
  - Optional Install for the Grid Infrastructure Management Repository 10-4
  - Support for DBMS_JOB 10-4
  - About Standard Edition High Availability 10-5
- Deprecated Features in Oracle Database 19c 10-5
  - Oracle Data Guard Broker Deprecated Properties 10-6
  - Oracle Data Guard Logical Standby Properties Deprecated 10-7
  - Deprecation of ASMCMD PWCREATE On Command Line 10-8
  - Deprecation of Addnode Script 10-8
  - Deprecation of clone.pl Script 10-8
  - Deprecation of Oracle Fail Safe 10-8
  - Deprecation of GDSCTL Operating System Command-Line Password Resets 10-8
  - Deprecation of Oracle Enterprise Manager Express 10-9
  - Deprecation of DV_REALM_OWNER Role 10-9
  - Deprecation of DV_REALM_RESOURCE Role 10-9
  - Deprecation of DV_PUBLIC Role 10-9
  - Deprecation of Oracle ACFS Replication Protocol REPV1 10-10
  - Deprecation of Oracle ACFS Encryption on Solaris and Windows 10-10
  - Deprecation of Oracle ACFS on Windows 10-10
  - Deprecation of Oracle ACFS Security (Vault) and ACFS Auditing 10-10
  - Deprecation of Oracle ACFS on Member Clusters (ACFS Remote) 10-10
  - Deprecation of Cluster Domain - Member Clusters 10-11
  - Deprecation of Vendor Clusterware Integration with Oracle Clusterware 10-11
  - Deprecation of Black Box Virtual Machine Management Using Oracle Clusterware 10-11
- Deprecated Initialization Parameters in Oracle Database 19c 10-11
  - CLUSTER_DATABASE_INSTANCES Initialization Parameter Deprecated 10-12
  - Deprecation of SQLNET.ENCRYPTION_WALLET_LOCATION Parameter 10-12
  - Deprecation of the SERVICE_NAMES Initialization Parameter 10-12
- Desupported Features in Oracle Database 19c 10-13
  - Desupport of Oracle Data Provider for .NET Promotable Transaction Setting 10-13
  - Desupport of Oracle Multimedia 10-14
  - Desupport of the CONTINUOUS_MINE feature of LogMiner 10-14

Desupport of Extended Datatype Support (EDS) 10-14
Data Guard Broker MaxConnections Property Desupported 10-14
Desupport of Leaf Nodes in Flex Cluster Architecture 10-15
Desupport of Oracle Streams 10-15
Desupport of PRODUCT_USER_PROFILE Table 10-15
Desupport of Oracle Real Application Clusters for Standard Edition 2 (SE2) Database Edition 10-15
Desupport of PDB Flat File Dictionary Dumps 10-16
Desupported Parameters in Oracle Database 19c 10-16
EXAFUSION_ENABLED Initialization Parameter Desupported 10-16
MAX_CONNECTIONS attribute of LOG_ARCHIVE_DEST_n Desupported 10-16
Desupport of O7_DICTIONARY_ACCESS 10-17
Desupport of OPTIMIZE_PROGRESS_TABLE Parameter 10-17
Behavior Changes, Deprecations and Desupports in Oracle Database 18c 10-17
Behavior Changes for Oracle Database 18c Upgrade Planning 10-18
Simplified Image-Based Oracle Database Installation 10-18
Support Indexing of JSON Key Names Longer Than 64 Characters 10-19
Upgrading Existing Databases is Replaced With Image Installations 10-19
About RPM-Based Oracle Database Installation 10-19
Token Limitations for Oracle Text Indexes 10-20
Changes to /ALL/USER/DBA User View and PL/SQL External Libraries 10-20
Symbolic Links and UTL_FILE 10-23
Deprecation of Direct Registration of Listeners with DBCA 10-24
UNIFORM_LOG_TIMESTAMP_FORMAT Changes in INIT.ORA 10-24
Deprecated Features in Oracle Database 18c 10-25
Data Guard MAX_CONNECTIONS Attribute is Deprecated 10-26
Extended Datatype Support (EDS) is Deprecated 10-26
GET_* Functions Deprecated in the DBMS_DATA_MINING Package 10-26
Package DBMS_XMLQUERY is deprecated 10-27
Package DBMS_XMLSAVE is Deprecated 10-27
Deprecated Columns in Oracle Label Security Views 10-27
Returning JSON True or False Values using NUMBER is Deprecated 10-28
Deprecation of MAIL_FILTER in Oracle Text 10-28
Deprecation of asmcmd showversion Option 10-28
Deprecation of NEWS_SECTION_GROUP in Oracle Text 10-28
Oracle Net Services Support for SDP is Deprecated 10-28
Deprecation of Flex Cluster (Hub/Leaf) Architecture 10-29
Deprecation of PRODUCT_USER_PROFILE Table 10-29
Desupported Features in Oracle Database 18c 10-29
Oracle Administration Assistant for Windows is Desupported 10-30
Oracle Multimedia DICOM Desupported Features 10-30

Oracle Multimedia Java Client Classes Desupported 10-30
Oracle XML DB Desupported Features 10-31
ODP.NET, Managed Driver - Distributed Transaction DLL Desupported 10-32
Data Guard Broker DGMGRL ALTER Syntax is Desupported 10-32
Desupport of CRSUSER on Microsoft Windows Systems 10-33
Desupported Initialization Parameters in Oracle Database 18c 10-33
Desupport of STANDBY_ARCHIVE_DEST Initialization Parameter 10-33
Desupport of UTL_FILE_DIR Initialization Parameter 10-33
Deprecation of Oracle Multimedia 10-34
Terminal Release of Oracle Streams 10-34
Behavior Changes, Deprecations and Desupports in Oracle Database 12c Release 2 (12.2) 10-34
Behavior Changes in Oracle Database 12c Release 2 (12.2) 10-35
Initialization Parameter Default Changes in Oracle Database 12c Release 2 (12.2) 10-36
Database Upgrade Assistant (DBUA) Enhancements and Changes 10-37
Enhancements to Oracle Data Guard Broker and Rolling Upgrades 10-38
About Changes in Default SGA Permissions for Oracle Database 10-38
Network Access Control Lists and Upgrade to Oracle Database 12c 10-39
Parallel Upgrade Utility Batch Scripts 10-40
Unified Auditing AUDIT_ADMIN and AUDIT_VIEWER Roles Changes 10-40
Oracle Update Batching Batch Size Settings Disabled 10-41
About Upgrading Tables Dependent on Oracle-Maintained Types 10-41
Case-Insensitive Passwords and ORA-1017 Invalid Username or Password 10-42
About Deploying Oracle Grid Infrastructure Using Oracle Fleet Patching and Provisioning 10-43
Restrictions Using Zero Data Loss Recovery Appliance Release 12.1 Backups 10-45
Client and Foreground Server Process Memory Changes 10-45
Deprecated Initialization Parameters in Oracle Database 12c Release 2 (12.2) 10-45
Deprecated Features in Oracle Database 12c Release 2 (12.2) 10-46
Deprecation of ALTER TYPE REPLACE 10-48
Deprecation of configToolAllCommands Script 10-48
Deprecation of DBMS_DEBUG Package 10-48
Deprecation of Intelligent Data Placement (IDC) 10-48
Deprecation of CONTINUOUS_MINE Option 10-49
Deprecation of Non-CDB Architecture 10-49
Deprecation of Oracle Administration Assistant for Windows 10-49
Deprecation of Oracle Data Provider for .NET PromotableTransaction Setting 10-49
Deprecation of oracle.jdbc.OracleConnection.unwrap() 10-50
Deprecation of oracle.jdbc.rowset Package 10-50

Deprecation of oracle.sql.DatumWithConnection Classes 10-50
Deprecation of Oracle Multimedia Java APIs 10-51
Deprecation of Oracle Multimedia Support for DICOM 10-51
Deprecation of Multimedia SQL/MM Still Image Standard Support 10-51
Deprecation of Unicode Collation Algorithm (UCA) 6.1 Collations 10-51
Deprecation of UNIFIED_AUDIT_SGA_QUEUE_SIZE 10-51
Deprecation of VERIFY_FUNCTION and VERIFY_FUNCTION_11G 10-52
Deprecation of V$MANAGED_STANDBY 10-52
Deprecation of Some XML DB Functions 10-52
Deprecated Features for Oracle XML Database 10-53
Desupported Initialization Parameters in Oracle Database 12c Release 2 (12.2) 10-55
Desupported Features in Oracle Database 12c Release 2 (12.2) 10-57
Desupport of Advanced Replication 10-57
Desupport of Direct File System Placement for OCR and Voting Files 10-58
Desupport of JPublisher 10-58
Desupport of preupgrd.sql and utluppkg.sql 10-58
Desupported Oracle Data Provider for .NET APIs for Transaction Guard 10-59
Desupported Views in Oracle Database 12c Release 2 (12.2) 10-59
SQLJ Support Inside Oracle Database 10-59
Desupport of Some XML DB Features 10-59
Desupport of MAX_SHARED_TEMP_SIZE 10-60
Behavior Changes, Deprecations and Desupports in Oracle Database 12c Release 1 (12.1) 10-60
Behavior Changes for Oracle Database 12c Release 1 (12.1) 10-61
Error Associated with catupgrd.sql Run Without PARALLEL=NO 10-62
Changes for Deinstallation and Cleanup of Oracle Base 10-62
Identifying and Dropping Deprecated and Desupported Parameters 10-63
Changes to Security Auditing Features 10-65
Upgrading a System that Did Not Have SQLNET.ALLOWED_LOGON_VERSION Parameter Setting 10-65
Oracle Warehouse Builder (OWB) Not Installed with Oracle Database 10-66
About Upgrading Oracle Database Release 10.2 or 11.1 and OCFS and RAW Devices 10-66
Change to VARCHAR2, NVARCHAR2, and RAW Datatypes 10-66
Changed Default for RESOURCE_LIMIT Parameter 10-67
Oracle XML DB is Mandatory and Cannot Be Uninstalled 10-67
Direct NFS Enabled By Default for Oracle RAC 10-67
Deprecated Features in Oracle Database 12c Release 1 (12.1) 10-68
Deprecation of Non-CDB Architecture 10-69
Deprecation of catupgrd.sql Script and Introduction of Parallel Upgrade Utility 10-70
DELETE_CATALOG_ROLE Deprecated 10-70

Deprecated Functions and Parameters in Oracle Label Security 10-70
Deprecation of IGNORECASE and SEC_CASE_SENSITIVE_LOGON 10-71
Deprecation of SQLNET.ALLOWED_LOGON_VERSION Parameter 10-71
Deprecation of Windows NTS Authentication Using the NTLM Protocol 10-72
Deprecation of Public Key Infrastructure for Transparent Data Encryption 10-72
Oracle Data Guard Broker Deprecated Features 10-73
Oracle Data Pump Export Utility Features 10-73
Deprecated Default Rule Sets for Oracle Database Vault 10-73
Deprecated Default Realms for Oracle Database Vault 10-74
Deprecated API for Oracle Database Vault 10-74
VPD Support in Oracle Database Semantic Technologies is Deprecated 10-74
Version-Enabled Models Support In Oracle Database Semantic Technologies 10-74
Deprecation of Oracle Streams 10-75
Deprecation of Advanced Replication 10-75
Deprecation of Single-Character SRVCTL CLI Options 10-76
Deprecation of Stored List of Administrative Users for Cluster Administration 10-76
Deprecation of -checkpasswd for QOSCTL Quality of Service (QoS) Command 10-76
Deprecation of SQLJ Inside the Server 10-76
Deprecated Oracle Update Batching 10-77
Deprecated EndToEndMetrics-related APIs 10-77
Deprecated Stored Outlines 10-77
Deprecated Concrete Classes in oracle.sql Package 10-78
Deprecated defineColumnType Method 10-78
Deprecated CONNECTION_PROPERTY_STREAM_CHUNK_SIZE Property 10-78
Deprecated Features for Oracle Call Interface 10-78
DICOM in Oracle Multimedia Deprecated in Oracle Database 12c Release 1 (12.1) 10-78
Deprecated NT LAN Manager (NTLM) Protocol for Oracle Net Services 10-78
Deprecated Views in Oracle Database 12c Release 1 (12.1) 10-79
Deprecated Initialization Parameters in Oracle Database 12c Release 1 (12.1) 10-79
Identifying and Dropping Deprecated and Desupported Parameters 10-80
FILE_MAPPING Initialization Parameter Deprecated 10-80
RDBMS_SERVER_DN Initialization Parameter Deprecated 10-81
SEC_CASE_SENSITIVE_LOGON Initialization Parameter Deprecated 10-81
Desupported Features in Oracle Database 12c Release 1(12.1) 10-81
Desupport of Oracle Enterprise Manager Database Control 10-83
Desupported Cipher Suites for Secure Sockets Layer (SSL) 10-84
Desupport of Database Rules Manager (RUL) and Expression Filter (EXF) 10-84

Desupport of Oracle COM Automation on Windows 10-85
Desupport of Oracle Objects for OLE 10-85
Desupport of Oracle Counters for Windows Performance Monitor 10-85
Programming Interface Deprecations for Oracle Data Provider for .NET 10-86
Desupport of Oracle Cluster File System (OCFS) on Windows 10-86
Desupport for Raw Storage Devices 10-86
Desupport of cluvfy comp cfs for OCFS 10-86
Desupport of Change Data Capture 10-87
Desupported Features in Oracle Data Mining 10-87
Desupported Implicit Connection Caching 10-87
Desupport of ABN Models for Oracle Data Mining Upgrades 10-88
Desupport of OLAP Catalog (AMD) 10-88
Desupport of CSSCAN and CSALTER for Oracle Globalization 10-89
Desupport of Oracle Net Connection Pooling 10-89
Desupport of Oracle Names 10-89
Desupport of Oracle Net Listener Password 10-90
Desupport of SQLNET.KERBEROS5_CONF_MIT Parameter for Oracle Net Services 10-90
Desupport of Oracle Names Control Utility for Oracle Net Services 10-90
Desupport of CTXXPATH in Oracle Text and Oracle XML DB 10-91
Desupport of ALTER INDEX OPTIMIZE for Text Indexes 10-91
Desupport of SYNC [MEMORY memsize] for Text Indexes 10-91

A Oracle Database Upgrade Utilities

Scripts for Upgrading Oracle Database A-1
Using the Pre-Upgrade Information Tool for Oracle Database A-4
About the Pre-Upgrade Information Tool A-5
Preupgrade Scripts Generated By the Pre-Upgrade Information Tool A-6
Postupgrade Scripts Generated By the Pre-Upgrade Information Tool A-7
Setting Up Environment Variables for the Pre-Upgrade Information Tool A-8
Pre-Upgrade Information Tool (preupgrade.jar) Command A-8
Output of the Pre-Upgrade Information Tool A-11
Pre-Upgrade Information Tool Output Example A-13
Pre-Upgrade Information Tool Warnings and Recommendations for Oracle Database A-21
Updating Access Control Lists and Network Utility Packages A-22
Evaluate Dependencies and Add ACLs for Network Utility Packages A-23
About Database Links with Passwords from Earlier Oracle Database Releases A-23

About Oracle Database Warnings for TIMESTAMP WITH TIME ZONE Data Type A-24

B Upgrading with Oracle Database Upgrade Assistant (DBUA)

Recommendations for Using DBUA B-1
About Stopping DBUA When Upgrading B-2
How DBUA Processes the Upgrade for Oracle Database B-3
Upgrade Scripts Started by DBUA B-3
Moving a Database from an Existing Oracle Home B-4
Using DBUA to Upgrade the Database on Linux, Unix, and Windows Systems B-6
Using DBUA in Silent Mode to Upgrade Oracle Database B-19
Running DBUA in Silent Mode B-19
DBUA Command-Line Syntax for Active and Silent Mode B-19

Index

List of Examples

2-1	Finding User Accounts That Use Case-Insensitive (10G) Version	2-22
2-2	Fixing Accounts with Case-Insensitive Passwords	2-23
2-3	Checking for the Presence of SEC_CASE_SENSITIVE_LOGON Set to FALSE	2-25
2-4	Using AutoUpgrade in Analyze Mode to Check an Oracle Database 12c Non-CDB System	2-35
2-5	AutoUpgrade Configuration File for a CDB and PDBs	2-37
2-6	Example of Running the Parallel Upgrade Utility using Priority List Emulation	2-40
2-7	Overview of Upgrade Procedure for Databases Using Oracle Data Vault	2-43
3-1	AutoUpgrade Configuration File for Non-CDB to PDB Conversion	3-14
3-2	Upgrading and Converting a Non-CDB to Oracle Database 19c Using Multitenant Architecture	3-61
3-3	Checking Redo Transport Service Status	3-65
3-4	Checking Apply Lag and Transport Lag	3-65
3-5		3-71
3-6	Setting up Archive Logging and Fast Recovery Area (FRA) Before Using AutoUpgrade	3-76
3-7	Password Files and Security Password File Updates	3-76
3-8	Transparent Data Encryption and AutoUpgrade	3-76
3-9	Example of How to Enable and Disable the AutoUpgrade Console	3-78
3-10	Starting Up the AutoUpgrade Utility in Analyze Mode	3-79
3-11	Creating a New Checklist for a Configuration File	3-79
3-12	Find and Edit checklist.cfg	3-81
4-1	Rerunning Upgrades With the Resume Option	4-6
4-2	Rerunning Upgrades With an Exclusion List	4-7
4-3	Rerunning Upgrades on PDBs Using the Resume Option	4-8
4-4	Rerunning Upgrades on PDBs Using Exclusion Lists	4-9
4-5	Rerunning Upgrades on PDBs Using an Inclusion List	4-10
4-6	Running Parallel Upgrade Utility with Parameters for CDB and Non-CDB Databases	4-20
4-7	Running Parallel Upgrades on Multiple Pluggable Databases (PDBs) Using Parallel Upgrade Utility	4-21
4-8	Specifying Complete PDB Upgrade Priority	4-33
4-9	Specifying a Priority Subset of PDBs, and Upgrading Other PDBs with Default Processing	4-34
4-10	Specifying a Priority Subset of PDBs, and Upgrading Other PDBs with an Inclusion List	4-34
4-11	Specifying a Priority Subset of PDBs, and Excluding CDB$ROOT with an Exclusion List	4-35
4-12	Specifying an Exclusion List using CATCTL_LISTONLY	4-36
4-13	Specifying a Priority List using CON_ID Values	4-36
4-14	Example of Multitenant Architecture Upgrade Using Defaults (No Parameters Set)	4-39

4-15	Example of Multitenant Architecture Upgrade Using 64 In Parallel PDB Upgrade Processors and 4 Parallel SQL Processes	4-39
4-16	Example of Multitenant Architecture Upgrade Using 20 In Parallel PDB Upgrade Processors and 2 Parallel SQL Processes	4-39
4-17	Example of Multitenant Architecture Upgrade Using 10 In Parallel PDB Upgrade Processors and 4 Parallel SQL Processes	4-39
4-18	Resetting the User Environment Variables	4-51
4-19	Testing the Connection to the Database	4-51
4-20	Running the Pre-Upgrade Information Tool (preupgrade.jar)	4-51
4-21	Reviewing the Pre-Upgrade Information Tool Log (preupgrade.log)	4-52
4-22	Running the Preupgrade Fixup SQL script (preupgrade_fixups.sql)	4-52
4-23	Manually Removing Oracle Enterprise Manager (em_present)	4-53
4-24	Manually Removing the OLAP Catalog (amd_exists)	4-54
4-25	Gathering Current Dictionary Statistics (dictionary_stats)	4-54
4-26	Granting the ADMINISTER DATABASE TRIGGER privilege (trgowner_no_admndbtrg)	4-55
4-27	Refreshing Materialized Views (mv_refresh)	4-55
4-28	Upgrading Oracle Application Express (apex_upgrade_msg)	4-56
4-29	Stopping the Database Service Using Command-Line Commands	4-58
4-30	Stopping the Database Service Using Microsoft Windows PowerShell Scripting	4-59
4-31	Deleting the Database Service from the Earlier Release Oracle Home	4-60
4-32	Stopping the Listener for the Earlier Release Oracle Home	4-60
4-33	Setting the environment variables to the new Oracle home	4-61
4-34	Copying the PFILE to the New Oracle home, and Creating a New Service Using the New Oracle Database binary	4-61
4-35	Starting the Database Upgrade	4-61
4-36	Completing the Post-Upgrade Checks	4-64
4-37	Running the Postupgrade Fixups Script	4-65
4-38	Manual Fixup Instructions for Oracle-Maintained Types In User Tables (depend_usr_tables)	4-66
4-39	Manual Fixup Instructions for Time Zone Version (old_time_zones_exist)	4-67
4-40	Manual Fixup for Refreshing Dictionary Statistics (post_dictionary)	4-72
5-1	Upgrade Summary Report for the Post-Upgrade Status Tool	5-11
6-1	Running the utlrp.sql Script On All Containers in the CDB With the CATCON Utility	6-6
6-2	Number of Invalid Objects Remaining	6-7
6-3	Number of Objects Recompiled	6-7
6-4	Number of Objects Recompiled with Errors	6-7

6-5	Copying the Earlier Release Demo Directory and Refreshing the Demonstrations in the Read-Only Oracle Home	6-14
6-6	Example of Spooling Postupgrade Fixup Results for a Non-CDB Oracle Database	6-20
6-7	Examples of Running Postupgrade Fixups Using catcon.pl On Linux, UNIX and Windows Systems	6-20
8-1	ORA-20001 Error Due To ORA-06512	8-31
10-1	Example of Error Messages with UTL_FILE And Symbolic Links	10-24
A-1	Non-CDB In the Source Oracle Home Example	A-11
A-2	CDB in a Source Oracle Home	A-11
B-1	Selecting a Database for Upgrade with DBUA	B-22
B-2	Selecting a Database for Upgrade with DBUA Using Noninteractive ("Silent") Option	B-22
B-3	Use Cases for Running DBUA in Noninteractive (”Silent") Mode	B-22

List of Figures

1-1 Upgrade Steps Workflow for Oracle Database 1-7

1-2 Example of an Oracle Database Release Number 1-12

2-1 Example of the Upgrade Checks Overview File 2-37

3-1 Converting and Upgrading a Non-CDB Using AutoUpgrade 3-14

4-1 Plug In a Non-CDB Using the DBMS_PDB.DESCRIBE Procedure 4-76

List of Tables

1-1	Examples of Upgrade Paths for Oracle Database 19c	1-4
1-2	The COMPATIBLE Initialization Parameter	1-16
1-3	Technology Methods for Migrating and Upgrading Databases During Upgrades	1-23
3-1	Global Configuration Parameters for Oracle Database AutoUpgrade Utility	3-37
3-2	Local Configuration Parameters for Oracle Database AutoUpgrade Utility	3-44
3-3	Locally Modifiable Global Parameters for AutoUpgrade Configuration Files	3-57
3-4	Internal Settings Configuration File Parameters for AutoUpgrade	3-71
4-1	Parallel Upgrade Utility (catctl.pl) Parameters	4-17
6-1	Common Oracle Database Vault Upgrade Scenarios and Upgrade Preparation Tasks	6-16
8-1	Supported Releases and Editions for Downgrading	8-2
10-1	Oracle Data Guard Property Name Changes	10-3
10-2	Deprecated columns in Oracle Label Security Views	10-27
A-1	Upgrade, Post-Upgrade, and Downgrade Scripts	A-1
A-2	Choices for Fixing the Time Zone File Version	A-25

Preface

These topics provide information about the scope of these contents for upgrading plans and procedures.

This book guides you through the process of planning and executing Oracle Database upgrades. In addition, this manual provides information about compatibility, upgrading applications, and important changes in the new Oracle Database release, such as initialization parameter changes and data dictionary changes.

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

- Audience
- Documentation Accessibility
- Set Up Java Access Bridge to Implement Java Accessibility
  Install Java Access Bridge so that assistive technologies on Microsoft Windows systems can use the Java Accessibility API.
- Related Documentation
- Conventions

Audience

Oracle Database Upgrade Guide is intended for database administrators (DBAs), application developers, security administrators, system operators, and anyone who plans or performs Oracle Database upgrades.

To use this document, you must be familiar with the following information:

- Relational database concepts
- Your current Oracle Database release
- Your operating system environment

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Set Up Java Access Bridge to Implement Java Accessibility

Install Java Access Bridge so that assistive technologies on Microsoft Windows systems can use the Java Accessibility API.

Java Access Bridge is a technology that enables Java applications and applets that implement the Java Accessibility API to be visible to assistive technologies on Microsoft Windows systems.

Refer to *Java Platform, Standard Edition Accessibility Guide* for information about the minimum supported versions of assistive technologies required to use Java Access Bridge. Also refer to this guide to obtain installation and testing instructions, and instructions for how to use Java Access Bridge.

Related Topics

- *Java Platform, Standard Edition Java Accessibility Guide*

Related Documentation

Review this documentation list for additional information.

- *Oracle Database Concepts* for a comprehensive introduction to the concepts and terminology used in this manual
- *Oracle Database Administrator's Guide* for information about administering Oracle Database
- *Oracle Database New Features Guide* for information about new features in this relaese
- *Oracle Database SQL Language Reference* for information on Oracle Database SQL commands and functions
- *Oracle Database Utilities* for information about utilities bundled with Oracle Database
- *Oracle Database Net Services Administrator's Guide* for information about Oracle Net Services

Many of the examples in this guide use the sample schemas, installed by default when you select the Basic Installation option with an Oracle Database installation. For information on how these schemas are created and how you can use them, refer to the following guide:

Oracle Database Sample Schemas

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction to Upgrading Oracle Database

Oracle provides upgrade options and strategies that are designed for your database environment, and an array of tools that automate the Oracle Database upgrade process.

- Overview of Oracle Database Upgrade Tools and Processes
  Review these topics to understand Oracle Database terms, tools and processes.
- Major Steps in the Upgrade Process for Oracle Database
  Oracle Database upgrades consist of six major steps.
- Compatibility and Interoperability Between Oracle Database Releases
  Learn how to understand and avoid compatibility and interoperability issues that can occur because of differences in Oracle Database releases.
- About Running Multiple Oracle Releases
  Run multiple releases using Optimal Flexible Architecture (OFA).
- About Converting Databases During Upgrades
  Review these topics to determine which is the best path for you to select to upgrade Oracle Databases.
- About Upgrading Platforms for a New Oracle Database Release
  Review these topics if you upgrade your operating system or hardware for a new Oracle Database release.
- About Image-Based Oracle Database Installation
  Starting with Oracle Database 18c, installation and configuration of Oracle Database software is simplified with image-based installation.

Overview of Oracle Database Upgrade Tools and Processes

Review these topics to understand Oracle Database terms, tools and processes.

- Definition of Terms Upgrading and Migrating
  Upgrading and migrating are different types of database changes.
- Upgrade and Data Migration Methods and Processes
  Oracle provides features and products to automate the upgrade process, and to assist you with completing upgrades efficiently.
- Oracle Database Releases That Support Direct Upgrade
  Review the supported options for direct upgrades to the latest Oracle Database release.
- Where to Find the Latest Information About Upgrading Oracle Database
  In addition to this document, Oracle provides information about upgrades on its support site, and through its Pre-Upgrade Information Tool.

Definition of Terms Upgrading and Migrating

Upgrading and migrating are different types of database changes.

Upgrading transforms an existing Oracle Database environment (including installed components and associated applications) into a new release Oracle Database environment. The data dictionary for the database is upgraded to the new release. Upgrading does not directly affect user data; no data is touched, changed, or moved during an upgrade.

Migrating data refers to moving data from one Oracle Database into another database previously created for migrating or moving the data. You migrate data when you need to move your database environment to a new hardware or operating system platform, or to a new character set. Migrating does not include upgrading to the latest release. The upgrade process is handled separately after you migrate the data.

The upgrade steps in *Oracle Database Upgrade Guide* apply to all operating systems, unless otherwise specified. Some operating systems can require additional upgrade steps.

Related Topics

- *Oracle Database Installation Guide*
- *Oracle Database Utilities*

Upgrade and Data Migration Methods and Processes

Oracle provides features and products to automate the upgrade process, and to assist you with completing upgrades efficiently.

Oracle Database supports the following methods for upgrading or migrating a database to the new release:

- Database Upgrade Assistant (DBUA)

  Provides a graphical user interface that guides you through the upgrade of a database. DBUA can be launched during installation with the Oracle Universal Installer, or you can launch DBUA as a standalone tool at any time in the future.

- Manual upgrade using the Parallel Upgrade Utility, and other command-line utilities

  Enables upgrades to be performed using shell scripts.

- Migrating data using Oracle Data Pump

  Provides export and import utilities. Oracle Data Pump can perform a full or partial export from your database, followed by a full, or partial import into the new release of Oracle Database. Export/Import in Oracle Data Pump can copy a subset of the data, leaving the database unchanged.

- `CREATE TABLE AS` SQL statement

  Migrates data from a database into a new Oracle Database release. By using this method, you can copy a subset of the data, leaving the database unchanged.

- Upgrading CDBs and PDBs using a priority list to group and upgrade PDBs according to their priority.

  Run the Parallel Upgrade Utility (`dbupgrade`, or `catctl.pl`) using the `-L` option to run the upgrade using a priority list, and to call that list as the upgrade runs.

- Synchronizing a standby database, upgrading, and using Oracle GoldenGate to synchronize the upgraded database (a zero downtime option)

- Using Fleet Patching and Provisioning (FPP) to upgrade databases.

In a Fleet Patching and Provisioning (FPP) upgrade (formerly known as Rapid Home Provisioning), you complete a new Oracle Database installation. After testing the database, and modifying it in accordance with the standard operating environment (SOE) that you want to use for your databases, you create an FPP gold image. A DBA deploys instances of that gold image to servers that have earlier release databases that you want to upgrade. After deployment of these gold images, a DBA can run a single `rhpctl` command to move files, perform configuration changes, and perform other steps required to use the new binaries. Refer to *Oracle Clusterware Administration and Deployment Guide* for more information about Rapid Home Provisioning.

Related Topics

- *Oracle Clusterware Administration and Deployment Guide*

Oracle Database Releases That Support Direct Upgrade

Review the supported options for direct upgrades to the latest Oracle Database release.

You can perform a direct upgrade to the new release from the following releases:

- 11.2.0.4
- 12.1.0.2
- 12.2.0.1
- 18

The path that you must take to upgrade to the latest Oracle Database release depends on the release number of your current database.

If your current Oracle Database is a release earlier than 11.2.0.4, then you cannot directly upgrade your Oracle Database to the latest release. In this case, you are required to upgrade to an intermediate release before upgrading to Oracle Database 19c.

If you cannot carry out a direct upgrade, then carry out an upgrade to the most recent release where direct upgrades are supported.

Note:

For any multi-step upgrade, if you must carry out two upgrades to upgrade to the current release, then you must run the preupgrade script twice: First, complete an upgrade to an intermediate upgrade release that is supported for direct upgrade to the target upgrade release. Second, complete the upgrade for the target upgrade release.

For example, if the database from which you are upgrading is running Oracle Database 10g, then to upgrade to Oracle Database 19c, follow these steps:

1. Upgrade release 10.2.0.5 to release 12.1.0.2 (the terminal patch set release), using the instructions in *Oracle Database Upgrade Guide 12c Release 1 (12.1)*, including running the preupgrade script for 12.1.0.2.
2. Upgrade Oracle Database 12c release 1 (12.1.0.2) directly to Oracle Database 19c. Use the instructions in this book, *Oracle Database Upgrade Guide*, including running the preupgrade script for Oracle Database 19c.

The following table shows the required upgrade path for each release of Oracle Database. Use the upgrade path and the specified documentation to perform an intermediate upgrade of your database before fully upgrading to Oracle Database 18c.

Table 1-1 Examples of Upgrade Paths for Oracle Database 19c

Current Release	Upgrade Options
18 (all releases), 12.2.0.1, 12.1.0.2, 11.2.0.4	**Direct upgrade is supported.** Perform the upgrade using the current Oracle Database Upgrade Guide, which is this guide.

Table 1-1 (Cont.) Examples of Upgrade Paths for Oracle Database 19c

Current Release	Upgrade Options
12.1.0.1 11.2.0.1, 11.2.0.2, 11.2.0.3 11.1.0.6, 11.1.0.7 10.2 or earlier releases	**Direct upgrade to Oracle Database 19c is not supported.** **Solution:** Upgrade to an intermediate Oracle Database release that can be directly upgraded to the current release. Upgrade Oracle Database releases that are not supported for direct upgrade in this release to an intermediate Oracle Database release that is supported for direct upgrade. When upgrading to an intermediate Oracle Database release, follow the instructions in the intermediate release documentation, including running the preupgrade scripts for that intermediate release. After you complete an upgrade to the intermediate release Oracle Database, you can upgrade the intermediate release database to the current Oracle Database release. This restriction does not apply if you use Oracle Data Pump export/import to migrate data to the new release. For example: • If you are upgrading from release 11.2.0.2 or 11.1.0.7, then you must first upgrade to the terminal patch set release for Oracle Database 11g release 2 (11.2.0.4), which is supported for direct upgrade to Oracle Database 19c. • If you are upgrading from release 10.2.0.2, 10.2.0.3, 10.2.0.4,10.2.0.5 or 10.1.0.5, then you must first upgrade to release 11.2. or 12.1 • If you are upgrading from release 9.2.0.8, then you must first upgrade to a sequence of intermediate Oracle Database releases: Upgrade from release 9.2.0.8 to release 11.2.0.4. Then upgrade from release 11.2 to 19c **Note:** Always update to the most recent intermediate release to which you can upgrade directly. Your case can be different from that of the examples provided here.

Where to Find the Latest Information About Upgrading Oracle Database

In addition to this document, Oracle provides information about upgrades on its support site, and through its Pre-Upgrade Information Tool.

Through its support website, Oracle provides late-breaking updates, discussions, and best practices about pre-upgrade requirements, upgrade processes, post-upgrade, compatibility, and interoperability.

Oracle also strongly recommends that you download and run the Pre-Upgrade Information Tool, which is available on My Oracle Support.

See Also:

- My Oracle Support:

  http://support.oracle.com

  You can search in the library, or search on My Oracle Support for keywords, such as "Database Upgrade"
- *Oracle Database Pre-Upgrade Utility* (Document ID 884522.1) at My Oracle Support

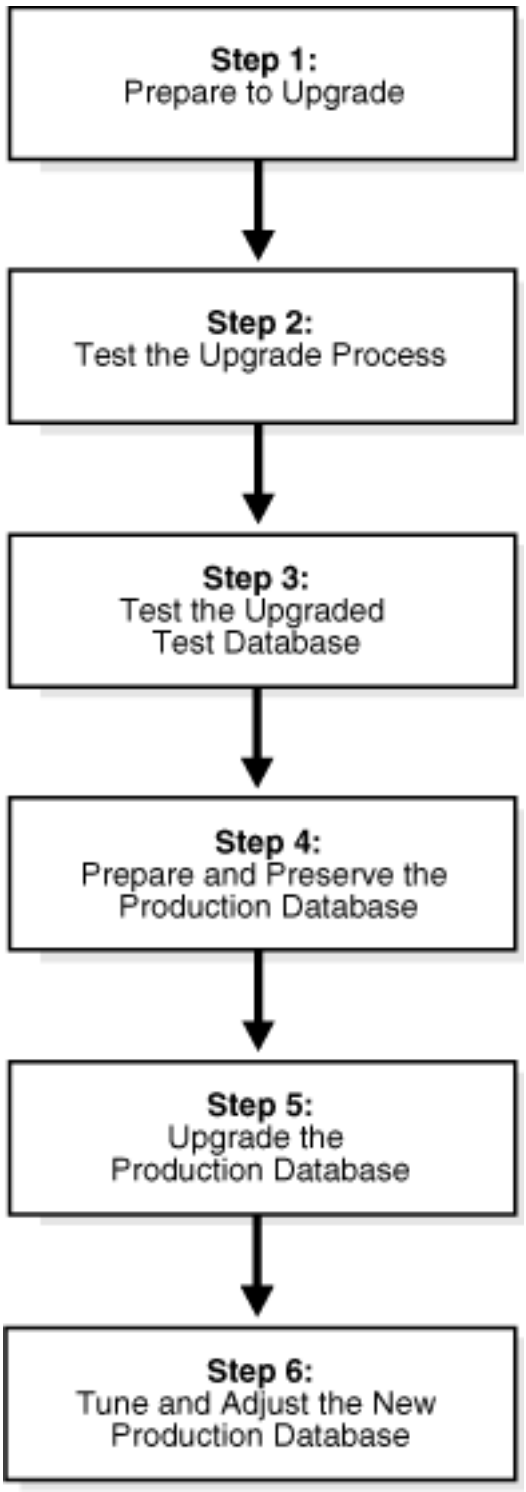
Major Steps in the Upgrade Process for Oracle Database

Oracle Database upgrades consist of six major steps.

Upgrade Steps Workflow

The following figure summarizes the major procedures performed during the upgrade process:

Figure 1-1 Upgrade Steps Workflow for Oracle Database



Step 1: Prepare to Upgrade Oracle Database

- Become familiar with the features of the new release of Oracle Database.
- Determine the upgrade path to the new release.

- Select an upgrade method.
- Select an Oracle home directory for the new release.
- Develop a testing plan.
- Prepare a backup strategy.
- Follow preupgrade recommendations.
- Run preupgrade fixup scripts, or carry out manual preupgrade system updates.

> **Note:**
>
> During the upgrade, consider running multiple releases of the database software, so that you can use the existing release as your production environment while you test the new release.
>
> Consider completing a software-only installation to the new Oracle Database release. In a software-only installation, you install the Oracle Database software but do not create a database as part of the installation process.

Step 2: Test the Upgrade Process for Oracle Database

- Perform a test upgrade using a test database. Conduct the test upgrade in an environment created for testing that does not interfere with the production database. Oracle recommends that your test environment is on a server that is, as much as possible, a replica of your production environment. For example: Oracle recommends that the server not only uses the same operating system, but that runs the same patch level, with the same packages, and matches other details of your production system configuration.

Step 3: Test the Upgraded Test Oracle Database

- Perform the tests that you planned in Step 1 on the test database that you upgraded to the new release of Oracle Database.
- Review the results, noting anomalies in the tests.
- Investigate ways to correct any anomalies that 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 successful and works with any required applications.
- To test for anomalies and determine potential support questions, carry out SQL plan management. SQL plan management includes the following steps:

  1. Before the upgrade, capture baselines and plans on the earlier release Oracle Database, and store those plans.

     Oracle recommends that you store the plans on staging tables, and then run the Data Pump Export utility `expdp` for those tables.

  2. After the upgrade, in the event of a regression or a performance issue, apply (load/accept/evolve) an old plan that you know is good, based on the plans you captured from the previous release Oracle Database.

See Also:

- *Oracle Database SQL Tuning Guide* for more information about SQL plan management
- Document 1948958.1 Patches to Consider for 11.2.0.3 to Avoid Problems with SQL Plan Management (SPM)
- Document 2034706.1 Patches to Consider for 11.2.0.4 to Avoid Problems with SQL Plan Management (SPM)
- Document 2035897.1 Patches to Consider When Upgrading From 12.1.0.1 to Avoid Problems with SQL Plan Management (SPM)

Step 4: Prepare and Preserve the Production Oracle Database

Complete these tasks before you upgrade your existing production database:

- Prepare the current production database as appropriate to ensure that the upgrade to the new release of Oracle Database is successful.
- Schedule the downtime required for backing up and upgrading the production database.
- Back up the current production database.

  Before you carry out a major change to a system, Oracle recommends that you make sure that you have a fallback strategy implemented. Oracle recommends that your fallback strategy includes the following preparations:

  – Test your backup strategy, and ensure that it works.
  – If you need a backup strategy, then plan for the time required to apply it during your maintenance window.
  – To perform plan stability checks in preparation for upgrade, carry out SQL plan management. Raise a service request if you need assistance.

Note:

A database upgrade that installs a new optimizer version usually results in plan changes for a small percentage of SQL statements.

Most plan changes result in either improvement or no performance change. However, some plan changes may cause performance regressions. SQL plan baselines significantly minimize potential regressions resulting from an upgrade.

When you upgrade, the database only uses plans from the plan baseline. The database puts new plans that are not in the current baseline into a holding area, and later evaluates them to determine whether they use fewer resources than the current plan in the baseline. If the plans perform better, then the database promotes them into the baseline; otherwise, the database does not promote them.

> **See Also:**
>
> *Oracle Database SQL Tuning Guide*

Step 5: Upgrade the Production Oracle Database

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

Step 6: Tune and Adjust the New Production Oracle Database

- Tune the new production database for the new release. Typically, the new production Oracle Database performs to the same standards, or better, than the database before the upgrade.
- Determine which features of the new Oracle Database release that you want to use, and update your applications accordingly.
- Develop new database administration procedures as needed.
- Do not upgrade your production Oracle Database release to the new release until all applications you must use in the upgraded database have been tested and operate properly.

Related Topics

- https://support.oracle.com/epmos/faces/DocumentDisplay?cmd=show&type=NOT&id=1948958.1
- https://support.oracle.com/epmos/faces/DocumentDisplay?cmd=show&type=NOT&id=2034706.1
- https://support.oracle.com/epmos/faces/DocumentDisplay?cmd=show&type=NOT&id=2035897.1

Compatibility and Interoperability Between Oracle Database Releases

Learn how to understand and avoid compatibility and interoperability issues that can occur because of differences in Oracle Database releases.

Oracle Database releases can have differences that can result in compatibility and interoperability issues. These differences can affect both general database administration and existing applications.

- About Oracle Database Release Numbers
  Oracle Database releases are categorized by five numeric segments that indicate release information.
- Convention for Referring to Release Numbers in Oracle Database Upgrade Guide
  Review to understand how statements apply to releases.
- What Is Oracle Database Compatibility?
  Before you upgrade, review compatibility between your earlier release Oracle Database and the new Oracle Database release as part of your upgrade plan.

- What Is Interoperability for Oracle Database Upgrades?
  In the context of upgrading Oracle Database, *interoperability* is the ability of different releases of Oracle Database to communicate and work in a distributed environment.
- About Invalid Schema Objects and Database Upgrades
  Run `utlrp.sql` to validate invalid objects as part of your upgrade test plan.
- About Upgrading Oracle OLAP Data Security Policies
  Security policies are stored differently starting with Oracle Database 12*c*. If your existing Oracle Database is 11*g* Release 11.2, then delete security roles.

About Oracle Database Release Numbers

Oracle Database releases are categorized by five numeric segments that indicate release information.

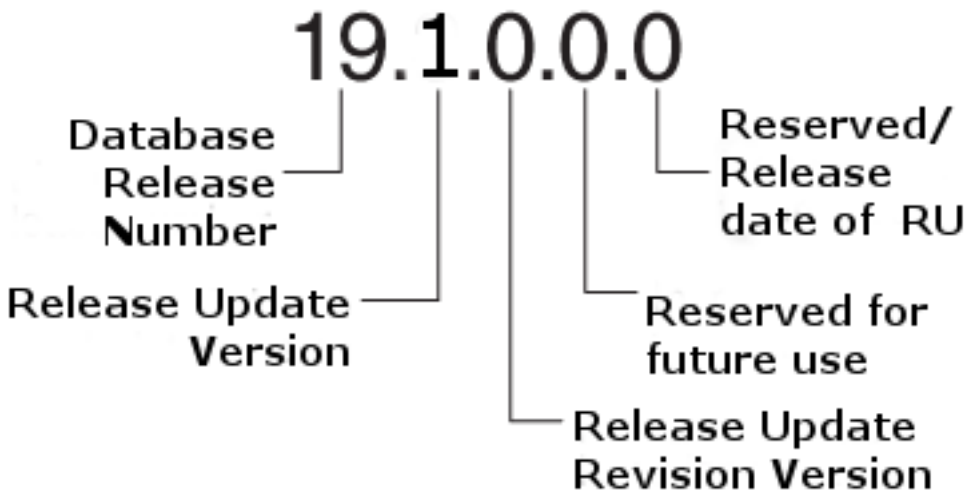
> **Note:**
>
> Starting with Oracle Database 18c, Oracle provides quarterly updates in the form of Release Updates (Updates) and Release Update Revisions (Revisions). Oracle no longer releases patch sets. For more information, see My Oracle Support Note 2285040.1.

Oracle Database releases are released in `version` and `version_full` releases.

The `version` release is designated in the form *`major release version`*`.0.0.0.0`. The major release version is based on the last two digits of the year in which an Oracle Database version is released for the first time. For example, the Oracle Database version released for the first time in the year 2018 has the major release version of 18, and thus its `version` release is `18.0.0.0.0`.

The `version_full` release is an update of a `version` release and is designated based on the major release version, the quarterly release update version (Update), and the quarterly release update revision version (Revision). The `version_full` releases are categorized by five numeric segments separated by periods as shown in the following example:

Figure 1-2 Example of an Oracle Database Release Number



- First numeral: This numeral indicates the major release version. It also denotes the last two digits of the year in which the Oracle Database version was released for the first time.
- Second numeral: This numeral indicates the release update version (Update).
- Third numeral: This numeral indicates the release update revision version (Revision).
- Fourth numeral: This numeral indicates the increment version. This nomenclature can apply to updates in future releases.
- Fifth numeral: This numeral is reserved for future use.

> **Note:**
>
> The first three numerals mainly identify an Oracle Database release.

> **Caution:**
>
> Oracle strongly recommends that you apply the most recent release update (Update) or bundle patch or patch set update to your source and target databases before starting an upgrade, and before starting a downgrade.

Related Topics

- My Oracle Support note 2285040.1

Convention for Referring to Release Numbers in *Oracle Database Upgrade Guide*

Review to understand how statements apply to releases.

When a statement is made in *Oracle Database Upgrade Guide* about a major database release number, the statement applies to all releases within that major database release.

Similarly, when a statement is made in *Oracle Database Upgrade Guide* about a maintenance release, the statement applies to all component-specific and platform-specific releases within that maintenance release. A statement about Oracle Database 12*c* applies to all component-specific and platform-specific releases within this release. A statement about Oracle Database 11*g* Release 2 (11.2) applies to release 11.2.0.2, release 11.2.0.3, and all other platform-specific releases within Oracle Database 11*g* Release 2 (11.2).

What Is Oracle Database Compatibility?

Before you upgrade, review compatibility between your earlier release Oracle Database and the new Oracle Database release as part of your upgrade plan.

- Understanding Oracle Database Compatibility
  If new features are incompatible with your earlier release, then Database compatibility can cause issues.
- When to Set the COMPATIBLE Initialization Parameter in Oracle Database
  Oracle recommends increasing the `COMPATIBLE` parameter only after you have completed testing the upgraded database.
- The COMPATIBLE Initialization Parameter in Oracle Database
  Review to understand how to set the COMPATIBLE initialization parameter for non-CDB and multitenant architecture containers.
- Values for the COMPATIBLE Initialization Parameter in Oracle Database
  Review to find the default, minimum, and maximum values for the `COMPATIBLE` initialization parameter for Oracle Database 19c.
- About Downgrading and Compatibility for Upgrading Oracle Database
  Before upgrading to Oracle Database 19c, you must set the `COMPATIBLE` initialization parameter to at least `11.2.0`.
- How the COMPATIBLE Initialization Parameter Operates in Oracle Database
  The `COMPATIBLE` initialization parameter enables or disables Oracle Database features based on release compatibility
- Checking the Compatibility Level of Oracle Database
  Use this SQL query to check that the compatibility level of your database corresponds to the value of the `COMPATIBLE` initialization parameter:

Understanding Oracle Database Compatibility

If new features are incompatible with your earlier release, then Database compatibility can cause issues.

Databases from different releases of Oracle Database software are compatible if they support the same features, and if those features perform the same way. When you upgrade to a new release of Oracle Database, certain new features can make your database incompatible with your earlier release.

Your upgraded database becomes incompatible with your earlier release under the following conditions:

- A new feature stores any data on disk (including data dictionary changes) that cannot be processed with your earlier release.
- An existing feature behaves differently in the new environment as compared to the old environment.

When to Set the COMPATIBLE Initialization Parameter in Oracle Database

Oracle recommends increasing the `COMPATIBLE` parameter only after you have completed testing the upgraded database.

After the upgrade is complete, you can increase the setting of the `COMPATIBLE` initialization parameter to the maximum level for the new Oracle Database release. However, after you increase the `COMPATIBLE` parameter, you cannot subsequently downgrade the database.

The COMPATIBLE Initialization Parameter in Oracle Database

Review to understand how to set the COMPATIBLE initialization parameter for non-CDB and multitenant architecture containers.

Oracle Database enables you to control the compatibility of your database with the `COMPATIBLE` initialization parameter.

Understanding the COMPATIBLE Initialization Parameter

In Oracle Database 19c, when the `COMPATIBLE` initialization parameter is not set in your parameter file, the COMPATIBLE parameter value defaults to `19.0.0` If you do not set the `COMPATIBLE` initialization parameter to `19.0.0`, then you cannot use the new Oracle Database 19c features, because your upgraded database is not running in the required `COMPATIBILITY` setting for Oracle Database 19c features.

When the Oracle Database `COMPATIBLE` parameter is increased to `19.0.0`, the first Java call to the database initiates a "name translation" operation. This operation can require a few minutes to complete. You should expect this delay the first time a Java call is made to the database after you increase the compatibility parameter. This initial delay to carry out the name translation occurs only during the initial Java call.

Note:

- Before upgrading to Oracle Database 19c, you must set the `COMPATIBLE` initialization parameter to at least `11.2.0`, which is the minimum setting for Oracle Database 19c.
- The compatible parameter must be at least 3 decimal numbers, separated by periods. For example:

```
SQL> ALTER SYSTEM SET COMPATIBLE = '12.1.0.2' SCOPE=SPFILE;
```

- Oracle recommends that you only raise the `COMPATIBLE` parameter after you have thoroughly tested the upgraded database.
- After you increase the `COMPATIBLE` parameter, you cannot downgrade the database.

Caution:

If you are upgrading from Oracle Database release 11.2, then you must set the compatible value to at least 11.2.0. You must do this at the time of the upgrade. Do not make this change until you are ready to upgrade, because a downgrade back to an earlier compatibility level is not possible after you raise the COMPATIBLE initialization parameter value.

See Also:

Oracle Database Administrator's Guide for information about managing initialization parameters

Rules for COMPATIBLE Parameter Settings in Multitenant Architecture

The `COMPATIBLE` parameter of the container database (CDB) affects the `COMPATIBLE` parameter settings of pluggable databases (PDBs) plugged into that container database. Review the following scenarios that occur when you plug in a PDB to a CDB:

- PDB `COMPATIBLE` equal to `CDB$ROOT` `COMPATIBLE` parameter setting.

  Result: No change to the PDB COMPATIBLE parameter setting.

- PDB `COMPATIBLE` is lower than `CDB$ROOT` `COMPATIBLE` parameter setting.

  Result: The PDB COMPATIBLE parameter is increased automatically to the same `COMPATIBLE` parameter setting as `CDB$ROOT`. After you plug in the PDB, you cannot downgrade the PDB to an earlier release.

- PDB `COMPATIBLE` is higher than `CDB$ROOT` `COMPATIBLE` parameter setting.

  Result: The PDB cannot be plugged in. Only PDBs with a `COMPATIBLE` parameter setting equal to or lower than `CDB$ROOT` can be plugged in to the CDB.

Values for the COMPATIBLE Initialization Parameter in Oracle Database

Review to find the default, minimum, and maximum values for the COMPATIBLE initialization parameter for Oracle Database 19c.

Default and Minimum COMPATIBLE Parameter Values

The COMPATIBLE parameter should not be changed for an RU or an RUR, either for CDB or Non-CDB instances. The following table lists the default and minimum values for the COMPATIBLE parameter in Oracle Database 19c, compared to earlier releases supported for direct upgrade:

Table 1-2 The COMPATIBLE Initialization Parameter

Oracle Database Release	Default Value	Minimum Value
Oracle Database 19c	19.0.0	11.2.0
Oracle Database 12c Release 2 (12.2)	12.2.0	11.2.0
Oracle Database 12c Release 1 (12.1)	12.0.0	11.0.0
Oracle Database 11g Release 2 (11.2)	11.2.0	10.0.0

About Downgrading and Compatibility for Upgrading Oracle Database

Before upgrading to Oracle Database 19c, you must set the COMPATIBLE initialization parameter to at least 11.2.0.

After upgrading to Oracle Database 18c, you can set the COMPATIBLE initialization parameter to match the release number of the new release. Doing so enables you to use all features of the new release, but prevents you from downgrading to your earlier release. Only a subset of Oracle Database 18c features are available while the COMPATIBLE initialization parameter is set to a lower value.

> **Caution:**
>
> After you increase the COMPATIBLE parameter, the database cannot be downgraded.

Related Topics

- Downgrading Oracle Database to an Earlier Release

How the COMPATIBLE Initialization Parameter Operates in Oracle Database

The COMPATIBLE initialization parameter enables or disables Oracle Database features based on release compatibility

The COMPATIBLE initialization parameter operates in the following way:

- The COMPATIBLE initialization parameter enables or disables the use of features, to help protect your existing application use of data.

If you run an Oracle Database 12*c* database with the `COMPATIBLE` initialization parameter set to `11.2.0`, then the database software generates database structures on disk that are compatible with Oracle Database Release 11*g* release 2 (11.2). If you try to use features that are part of a later release of Oracle Database, and make the database incompatible with the `COMPATIBLE` initialization parameter, then an error occurs. However, new features are enabled that do not create changes on disk that are incompatible with Oracle Database Release 11*g* release 2.

- If you make changes to the database that make the database incompatible with the `COMPATIBLE` initialization parameter setting you want to use, then the database does not start, and initialization terminates in an error. If this happens, then you must set the `COMPATIBLE` initialization parameter to an appropriate value for the database.

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

Checking the Compatibility Level of Oracle Database

Use this SQL query to check that the compatibility level of your database corresponds to the value of the `COMPATIBLE` initialization parameter:

```
SQL> SELECT name, value FROM v$parameter
         WHERE name = 'compatible';
```

What Is Interoperability for Oracle Database Upgrades?

In the context of upgrading Oracle Database, *interoperability* is the ability of different releases of Oracle Database to communicate and work in a distributed environment.

A distributed database system can comprise different releases of Oracle Database, and all supported releases of Oracle Database can participate in the distributed database system. However, the applications that work with a distributed database must also be able to interoperate with the features and functions that are available at each node in the system.

Interoperability across disparate operating systems and operating system versions can cause problems (especially during rolling upgrades) because the minimum requirements for the new Oracle Database release may require you to upgrade the operating systems on some or all of your hosts. For this reason, before you start an Oracle Database upgrade, you must check to ensure that drivers, network, and storage are compatible for all the interim upgrade states of the system during the rolling upgrade.

> **Note:**
>
> Because *Oracle Database Upgrade Guide* discusses upgrading and downgrading between different releases of Oracle Database, the definition of *interoperability* is for Oracle Database releases. Other Oracle documentation may use a broader definition of the term *interoperability*. For example, interoperability in some cases can describe communication between different hardware platforms and operating systems.
>
> My Oracle Support note 207303.1 "Client / Server / Interoperability Support Between Different Oracle Versions" provides additional information.

Related Topics

- https://support.oracle.com/rs?type=doc&id=207303.1

About Invalid Schema Objects and Database Upgrades

Run `utlrp.sql` to validate invalid objects as part of your upgrade test plan.

After database upgrades, release changes can result in invalid schema objects in the upgraded database. Typically, invalid objects fix themselves as they are accessed or run. However, Oracle recommends that you recompile invalid objects in the database as part of your patching and upgrade procedure, so that you resolve issues with invalid objects, and any required dependencies, before users encounter these invalid objects.

Object validation is an operation that checks the Oracle Database Data Definition Language (DDL) statements. These statements are used to define the database structure or schema. Validating DDL statements can take time to complete. The following is a list of some common factors that can affect object validation time:

- Number of invalid objects
- CPU types
- Processor speeds
- System loads
- Available physical memory

The `utlrp.sql` command recompiles all objects in an invalid state, including packages, procedures, and types. It is located in the `$ORACLE_HOME/rdbms/admin` directory. The `utlrp.sql` script automatically runs in serial or in parallel recompilation, based on the number of CPUs available (identified by the parameter `cpu_count`), multiplied by the number of threads for each CPU (identified by the parameter `parallel_threads_per_cpu`). On Oracle Real Application Clusters systems (Oracle RAC), the number of parallel threads is added across all Oracle RAC nodes.

Run the command either as the SYS user, or as another user account that is granted the SYSDBA system privileges.

Oracle recommends that you run the `utlrp.sql` command in the earlier release Oracle Database to recompile any existing invalid objects in your database. Particularly ensure that SYS and SYSTEM user schema invalid objects are updated. During upgrade tests, run `utlrp.sql` in the upgraded Oracle Database as part of

your upgrade test plan, so that you can include planning for recompilation time as part of your upgrade. Recompilation time is proportional to the number of invalid objects in the database. If the upgrade results in a large number of invalid objects, then `utlrp.sql` can take a significant amount of time to run.

About Upgrading Oracle OLAP Data Security Policies

Security policies are stored differently starting with Oracle Database 12*c*. If your existing Oracle Database is 11*g* Release 11.2, then delete security roles.

In Oracle Database 12*c*, Oracle OLAP uses Oracle Real Application Security (ORAS) to store OLAP data security policies instead of Extensible Data Security (XDS), which it used in Oracle Database 11*g* releases.

When you upgrade Oracle Database from release 11*g* to new Oracle Database releases, any XDS data security policies are automatically converted to ORAS.

> **Note:**
>
> Data security roles defined in a release 11*g* Oracle Database instance are not automatically converted to ORAS. Before you upgrade Oracle Database Release 11*g* to a current Oracle Database release, you must delete any data security roles that are defined in the 11*g* database. After the upgrade, you can use the new release Oracle Database Analytic Workspace Manager to define the data security roles again.
>
> If you upgrade an 11*g* database without deleting the 11*g* data security roles, then any data security policies that include a data security role are invalid in the later Oracle Database releases.

Related Topics

- *Oracle OLAP User's Guide*

About Running Multiple Oracle Releases

Run multiple releases using Optimal Flexible Architecture (OFA).

Optimal Flexible Architecture (OFA) is a set of configuration guidelines for efficient and reliable Oracle Database and Oracle Grid Infrastructure deployments. Oracle recommends that you deploy all Oracle software installations in accordance with the OFA architecture standard for Oracle Database installations. Following the OFA standard helps to ensure that your installations are easier for you to maintain, and easier for you to obtain rapid assistance from Oracle Support.

OFA provides the following benefits:

- Organizes large amounts of complicated software and data on disk, which can help 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
- Simplifies the administration of multiple Oracle databases

- Helps eliminate fragmentation of free space in the data dictionary, isolates other fragmentation, and helps to minimize resource contention
- Assists database administrators to deploy an effective enterprise data management strategy

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.

For more information about OFA, refer to your operating system-specific Oracle documentation. For more information about managing data files and temp files, refer to *Oracle Database Administrator's Guide*.

- Databases in Multiple Oracle Homes on Separate Computers
  Review if you want to connect to multiple releases using Oracle Database clients.
- Databases in Multiple Oracle Homes on the Same Computer
  Installing earlier releases of Oracle Database on the same computer that is running Oracle Database 19c can cause issues with client connections.
- About the Optimal Flexible Architecture Standard
  Oracle Optimal Flexible Architecture (OFA) rules help you to organize database software and configure databases to allow multiple databases, of different versions, owned by different users to coexist.
- About Multiple Oracle Homes Support
  Oracle Database supports multiple Oracle homes. You can install this release or earlier releases of the software more than once on the same system, in different Oracle home directories.

Related Topics

- *Oracle Database Administrator's Guide*

Databases in Multiple Oracle Homes on Separate Computers

Review if you want to connect to multiple releases using Oracle Database clients.

You can install Oracle Database 12*c* and Oracle Database 11*g* databases in Oracle homes on separate computers and Oracle Database 12*c* with Oracle Database 11*g* clients connecting to any or all of the databases. However, you must install the latest release first and install earlier releases subsequently in descending chronological order. Installing in descending chronological order ensures that each installation can find the Oracle inventory and register its installation, so that you can avoid a corruption of the Oracle inventory.

Databases in Multiple Oracle Homes on the Same Computer

Installing earlier releases of Oracle Database on the same computer that is running Oracle Database 19c can cause issues with client connections.

You can find that you are not able to install earlier releases of Oracle Database on the same computer that is running Oracle Database 18c, and then have clients connect to the databases of the earlier releases. For example, you cannot have Oracle Database 11g, Oracle Database 12c, Oracle Database 18c and Oracle Database 19c databases in multiple (or separate) Oracle homes on the same computer, and have Oracle Database 11c, Oracle Database 12c, Oracle Database 18c and Oracle Database 19c clients connecting to any or all of the databases on this computer. It is possible to have a combination of some releases on one system.

Oracle recommends that you obtain the latest information about compatibility and supported configurations from My Oracle Support Note 207303.1 "Client / Server / Interoperability Support Between Different Oracle Versions" on My Oracle Support.

> **See Also:**
>
> https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=207303.1

About the Optimal Flexible Architecture Standard

Oracle Optimal Flexible Architecture (OFA) rules help you to organize database software and configure databases to allow multiple databases, of different versions, owned by different users to coexist.

In earlier Oracle Database releases, the OFA rules provided optimal system performance by isolating fragmentation and minimizing contention. In current releases, OFA rules provide consistency in database management and support, and simplifies expanding or adding databases, or adding additional hardware.

By default, Oracle Universal Installer places Oracle Database components in directory locations and with permissions in compliance with OFA rules. Oracle recommends that you configure all Oracle components on the installation media in accordance with OFA guidelines.

Oracle recommends that you accept the OFA default. Following OFA rules is especially of value if the database is large, or if you plan to have multiple databases.

> **Note:**
>
> OFA assists in identification of an ORACLE_BASE with its Automatic Diagnostic Repository (ADR) diagnostic data to properly collect incidents.

About Multiple Oracle Homes Support

Oracle Database supports multiple Oracle homes. You can install this release or earlier releases of the software more than once on the same system, in different Oracle home directories.

Careful selection of mount point names can make Oracle software easier to administer. Configuring multiple Oracle homes in compliance with Optimal Flexible Architecture (OFA) rules provides the following advantages:

- You can install this release, or earlier releases of the software, more than once on the same system, in different Oracle home directories. However, you cannot install products from one release of Oracle Database into an Oracle home directory of a different release. For example, you cannot install Oracle Database 19c software into an existing Oracle 18c Oracle home directory.

- Multiple databases, of different versions, owned by different users can coexist concurrently.
- You must install a new Oracle Database release in a new Oracle home that is separate from earlier releases of Oracle Database.

  You cannot install multiple releases in one Oracle home. Oracle recommends that you create a separate Oracle Database Oracle home for each release, in accordance with the Optimal Flexible Architecture (OFA) guidelines.
- In production, the Oracle Database server software release must be the same as the Oracle Database dictionary release through the first four digits (the major, maintenance, and patch release number).
- Later Oracle Database releases can access earlier Oracle Database releases. However, this access is only for upgrades. For example, Oracle Database 19c can access an Oracle Database 18c if the 18c database is started up in upgrade mode.
- Oracle Database Client can be installed in the same Oracle Database home if both products are at the same release level. For example, you can install Oracle Database Client 12.2.0.1 into an existing Oracle Database 12.2.0.1 home but you cannot install Oracle Database Client 12.2.0.1 into an existing Oracle Database 12.1.0.2 home. If you apply a patch set before installing the client, then you must apply the patch set again.
- Structured organization of directories and files, and consistent naming for database files simplify database administration.
- Login home directories are not at risk when database administrators add, move, or delete Oracle home directories.
- You can test software upgrades in an Oracle home in a separate directory from the Oracle home where your production database is located.

About Converting Databases During Upgrades

Review these topics to determine which is the best path for you to select to upgrade Oracle Databases.

- Overview of Converting Databases During Upgrades
  There are two methods to convert non-CDBs to multitenant architecture Oracle Databases during upgrades, and several different technologies you can use.
- About Upgrading Using Standby Databases
  You can perform rolling upgrades of databases by using Active Oracle Data Guard, or by using Oracle Enterprise Manager Cloud Control.
- Migrating From Standard Edition to Enterprise Edition of Oracle Database
  Review these options to migrate to Oracle Database Enterprise Edition from Oracle Database Standard Edition
- Migrating from Enterprise Edition to Standard Edition of Oracle Database
  Converting from Enterprise Edition to Standard Edition requires exporting and importing data.
- Migrating from Oracle Database Express Edition (Oracle Database XE) to Oracle Database
  You must upgrade from Oracle Database Express Edition to Oracle Database Enterprise Edition, and then upgrade to the current Oracle Database release.

Overview of Converting Databases During Upgrades

There are two methods to convert non-CDBs to multitenant architecture Oracle Databases during upgrades, and several different technologies you can use.

Oracle Database 19c is the terminal release for non-CDB Oracle Database upgrades to non-CDB architecture. Review the upgrade options for migration and upgrade from non-CDB Oracle Database architecture to Oracle Database deployed using the multitenant architecture.

Options for Manual Migration and Upgrades of Oracle Database Non-CDB Architecture to the Multitenant Architecture

There are two ways you can perform manual migrations and upgrades of non-CDBs to Oracle Database container databases (CDBs) and pluggable databases (PDBs), which use the multitenant architecture:

- Convert the non-CDB to a PDB before upgrade.

  With this option, you plug in the non-CDB Oracle Database release to the same release CDB. (For example, plug in a non-CDB Oracle Database Release 18c into an Oracle Database 18c release CDB). Finish converting the non-CDB Oracle Database to a PDB. Then, upgrade the entire CDB, with its PDBs, to Oracle Database 19c.

- Plug in the non-CDB, upgrade, and finish converting the non-CDB to a PDB after upgrade.

  With this option, you plug in a non-CDB Oracle Database release to an Oracle Database 19c CDB. Upgrade the plugged-in non-CDB Oracle Database to Oracle Database 19c. Then, finish converting the non-CDB Oracle Database to a PDB.

Upgrade Technology Methods for Options

The following table lists methods that you can use to convert upgrades, including references to availability issues. It also provides references to the documentation that describes how to carry out each upgrade method.

Table 1-3 Technology Methods for Migrating and Upgrading Databases During Upgrades

Method	Description	Reference
AutoUpgrade for Oracle Database	Use the AutoUpgrade tool to automate the upgrade of your database.	Refer to the topic "Using AutoUpgrade for Oracle Database Upgrades"
Oracle Data Guard Transient Standby (Physical Standby) database	Use an existing physical standby database to perform a database upgrade by temporarily converting it to a logical standby database, and then converting it back to a physical standby.	Refer to the topic “About Upgrading Using Standby Databases”
Parallel Upgrade Utility	Use the Parallel Upgrade Utility to perform a manual upgrade of Oracle Database. If the Oracle Database is a non-CDB, then you must convert the database to a PDB	Refer to the topics under "Upgrading Oracle Database with DBUA or Parallel Upgrade Utility"

Table 1-3 (Cont.) Technology Methods for Migrating and Upgrading Databases During Upgrades

Method	Description	Reference
Oracle GoldenGate synchronization of production and standby databases for zero downtime upgrades	Use Oracle GoldenGate with software upgrades and with Oracle Database data migration procedures to carry out a synchronization approach to maintaining availability during an upgrade. • Use RMAN restore and upgrade to set up a standby database running the earlier release software using an existing backup • Upgrade the standby database to the new Oracle Database release • Move the entire database and synchronize the standby database with the production database using the following tools: – Oracle Data Pump – Transportable Tablespaces (TTS) – `CREATE TABLE AS SELECT` (CTIS) to create new tables and populate them with rows from specified queries. – `INSERT AS SELECT` (IAS) to create nonpartitioned tables • Use Data Load/Unload to load data into the new database release, and unload data from the old database release	Refer to Oracle GoldenGate documentation, and relevant Oracle Database documentation for your use case.
Oracle Enterprise Manager Cloud Control	Oracle provides Cloud Control support for performing database upgrades with Oracle Database 12c and later releases. This option requires that you purchase the Enterprise Manager Lifecycle Management Pack.	Refer to Online help in Oracle Enterprise Manager Cloud Control

> **Note:**
>
> Upgrades of Oracle Grid Infrastructure (Oracle Clusterware and Oracle Automatic Storage Management) are carried out separately, before Oracle Database upgrades. You must complete Oracle Grid Infrastructure upgrades before you upgrade Oracle Database installations. Other features installed with Oracle Database can have additional upgrade requirements.

Related Topics

- *Oracle Grid Infrastructure Installation Guide* for your platform
- Oracle GoldenGate documentation

About Upgrading Using Standby Databases

You can perform rolling upgrades of databases by using Active Oracle Data Guard, or by using Oracle Enterprise Manager Cloud Control.

The `DBMS_ROLLING` PL/SQL package enables you to upgrade the database software in an Oracle Data Guard configuration in a rolling fashion. Rolling upgrades using Active Data Guard uses an Oracle Data Guard physical standby database and the SQL Apply process. Using Data Guard for rolling upgrades is supported for Oracle Database 12c release 1 (12.1) and later Oracle Database releases.

With Oracle Database 12c release 2 (12.2) and later releases, when you perform a rolling upgrade using the DBMS_ROLLING PL/SQL package, you no longer need to disable the broker. In addition, the broker now reports when a rolling upgrade is in place, and tracks its status. The status information is displayed in the output of the DGMGRL commands SHOW CONFIGURATION and SHOW DATABASE.

Oracle Enterprise Manager Cloud Control provides options to perform a rolling upgrade of databases in a Data Guard configuration. The procedures are described in online help within Cloud Control.

See Also:

- *Oracle Data Guard Broker* for information about upgrading and downgrading in an Oracle Data Guard broker configuration
- *Oracle Data Guard Concepts and Administration* for information about using `DBMS_ROLLING` to perform a rolling upgrade.

Migrating From Standard Edition to Enterprise Edition of Oracle Database

Review these options to migrate to Oracle Database Enterprise Edition from Oracle Database Standard Edition

If you have Oracle Database Standard Edition at a release earlier than the new Oracle Database release, then you can change it from a Standard Edition release to Oracle Database Enterprise Edition by selecting one of the following options:

- Perform a normal upgrade procedure.

  Install Oracle Enterprise Edition software in a new Oracle home, and follow the normal upgrade procedures as described in the "Upgrading Oracle Database" chapter. The Data Dictionary for Standard Edition and Enterprise Edition configurations are the same. The difference between Standard Edition and Enterprise Edition is in the options that are available in the server executable.
- Perform an In-Place Upgrade using the same Oracle home.

  If you have a Standard Edition database at a release earlier than the new release of Oracle Database, and you want to perform an in-place upgrade using the same Oracle home, then you must first upgrade the Standard Edition Database. After

you complete the upgrade, use the procedure described here to install Oracle Database Enterprise Edition software and to move to Oracle Database Enterprise Edition.

> **Caution:**
>
> Performing this procedure deinstalls the Oracle Standard Edition software. It results in deleting database files that exist under the Oracle home, and under the Fast Recovery Area (FRA). Back up database files under the current Oracle home before you begin this procedure.

1. Ensure that the release number of your Oracle Standard Edition server software is the same release as your Oracle Enterprise Edition server software.
2. Shut down your database.
3. If your operating system is Windows, then stop all Oracle services, including the `OracleService`*`SID`* Oracle service, where *`SID`* is the instance name.
4. Back up all database files under the current Oracle home that you must keep.
5. Deinstall the Standard Edition server software.

   > **Caution:**
   >
   > This step deletes all existing database files that reside under the Oracle home.

   Run the deinstallation tool from the Oracle home. The deinstallation tool is available as a separate command (`deinstall`) under the Oracle home directory after installation. It is located under *`ORACLE_HOME`*`\deinstall`.

   To deinstall an Oracle home on Windows, use the following syntax:

   ```
   setup.exe –deinstall –home path_of_Oracle_home_to_be_deinstalled
   ```

   To deinstall an Oracle home on Linux and UNIX, use the following syntax:

   ```
   $ ./runInstaller –deinstall –home
   path_of_Oracle_home_to_be_deinstalled
   ```

> **Note:**
>
> Starting with Oracle Database 12*c*, the deinstallation tool is integrated with the database installation media. You can run the deinstallation tool using `runInstaller` on Linux and UNIX, or by using `setup.exe` on Windows with the `-deinstall` and `-home` options from the base directory of the Oracle Database, Oracle Database Client, or Oracle Grid Infrastructure installation media.

6. Install Oracle Enterprise Edition server software using Oracle Universal Installer (OUI).

   Select the same Oracle home that was used for the Standard Edition that you uninstalled, or select a new Oracle home. During the installation, be sure to select Enterprise Edition. When prompted, choose `Software Only` from the Database Configuration screen.

7. If you have an existing database, then set your *`ORACLE_SID`* to this preexisting database.

   If your existing database is on Windows, then you must recreate the database service by using the `ORADIM` utility.

8. Start up your database.

Related Topics

- Upgrading Oracle Database
  You can upgrade manually by using the Parallel Upgrade Utility command-line option.

Migrating from Enterprise Edition to Standard Edition of Oracle Database

Converting from Enterprise Edition to Standard Edition requires exporting and importing data.

To properly convert from an Enterprise Edition database to a Standard Edition database, you must perform an Export/Import operation. Oracle recommends that you use the Standard Edition Export utility to export the data. If you only install Standard Edition software, then some data dictionary objects become invalid. These invalid objects create problems when maintaining the database.

The Export/Import operation does not introduce data dictionary objects specific to the Enterprise Edition, because the `SYS` schema objects are not exported. After the Import in the Standard Edition database, you are only required to drop user schemas related to Enterprise Edition features.

Migrating from Oracle Database Express Edition (Oracle Database XE) to Oracle Database

You must upgrade from Oracle Database Express Edition to Oracle Database Enterprise Edition, and then upgrade to the current Oracle Database release.

Oracle Database Express Edition (Oracle Database XE) is an entry-level edition of Oracle Database.

To upgrade Oracle Database 11g release 2 (11.2) Express Edition (Oracle Database XE) to Oracle Database 12c Release 2 or later releases, you must first upgrade from Oracle Database XE to Oracle Database 12c Release 1 (12.1.0.2) Enterprise Edition, and then upgrade to a later Oracle Database Enterprise Edition release.

For more information, see the "Oracle Database Express Edition (XE)" Oracle online forum:

`http://forums.oracle.com`

About Upgrading Platforms for a New Oracle Database Release

Review these topics if you upgrade your operating system or hardware for a new Oracle Database release.

- About Upgrading Your Operating System
  Check operating system requirements for new releases, and if necessary, upgrade your operating system before upgrading Oracle Database.
- Options for Transporting Data to a Different Operating System
  Review these restrictions and guidelines if you want to perform a cross-platform upgrade.

About Upgrading Your Operating System

Check operating system requirements for new releases, and if necessary, upgrade your operating system before upgrading Oracle Database.

When you upgrade to a new release of Oracle software, the operating system requirements may have changed. If required, upgrade the operating system before upgrading Oracle Database.

See Also:

- *Oracle Database Installation Guide* for your platform to obtain a list of supported operating systems
- Your operating system-specific documentation for information about how to perform an operating system upgrade

Options for Transporting Data to a Different Operating System

Review these restrictions and guidelines if you want to perform a cross-platform upgrade.

When using DBUA or when performing a manual upgrade for Oracle Database, you cannot directly migrate or transport data in a database on one operating system to a

database on another operating system. For example, you cannot migrate data in an Oracle database on Solaris to an Oracle 12*c* database on Windows using DBUA. You must follow procedures specific to your operating system platforms.

To see the platforms that support cross-platform data transport, run the following query using SQL*Plus:

```
SELECT * FROM V$TRANSPORTABLE_PLATFORM ORDER BY PLATFORM_NAME;
```

Note:

If the source platform and the target platform are of different endianness, then you cannot use the `RMAN CONVERT DATABASE` command. This process requires both the source and target platform to be the same endian value. Your available options are Data Pump replication, Data Pump export/import, or Transportable Tablespace, with an `RMAN CONVERT TABLESPACE`. If the platforms are of the same endianness, then no conversion is necessary and data can be transported as if on the same platform.

See Also:

- *Oracle Database Administrator's Guide* for a discussion of transporting data across platforms
- *Oracle Database Backup and Recovery User's Guide* for information on using the `RMAN CONVERT DATABASE` and `RMAN CONVERT TABLESPACE` commands

About Image-Based Oracle Database Installation

Starting with Oracle Database 18c, installation and configuration of Oracle Database software is simplified with image-based installation.

To install Oracle Database, create the new Oracle home, extract the image file into the newly-created Oracle home, and run the setup wizard to register the Oracle Database product.

Using image-based installation, you can install and upgrade Oracle Database for single-instance and cluster configurations.

This installation feature streamlines the installation process and supports automation of large-scale custom deployments. You can also use this installation method for deployment of customized images, after you patch the base-release software with the necessary Release Updates (Updates) or Release Update Revisions (Revisions).

Note:

You must extract the image software (`db_home.zip`) into the directory where you want your Oracle Database home to be located, and then run the Oracle Database Setup Wizard to start the Oracle Database installation and configuration. Oracle recommends that the Oracle home directory path you create is in compliance with the Oracle Optimal Flexible Architecture recommendations.

2
Preparing to Upgrade Oracle Database

Complete preupgrade tasks and checks to assist you with completing a successful upgrade.

This chapter provides information and procedures for the pre-upgrade tasks, including planning your upgrades, data-gathering, testing, installing the new Oracle software for the upgrade, using the Parallel Upgrade Utility to carry out your upgrade, and performing other checks and tasks.

- Tasks to Prepare for Oracle Database Upgrades
  Carry out these tasks to prepare your upgrade.
- Pre-Upgrade Information Tool Checklist
  To obtain a checklist of tasks you must complete before upgrading an Oracle Database on a physical server or virtual machine, run the Pre-Upgrade Information Tool (`preupgrade.jar`).
- Installing the New Oracle Database Software for Single Instance
  Use this procedure overview to assist you to install the software for the new Oracle Database release for a single instance deployment.
- Installing the New Oracle Database Software for Oracle RAC
  Use this procedure overview to assist you to install the software for the new Oracle Database release for an Oracle RAC deployment.
- Database Preparation Tasks to Complete Before Starting Oracle Database Upgrades
  Ensure that you have completed these database preparation tasks before starting an Oracle Database upgrade.
- Preparing for Database Rolling Upgrades Using Oracle Data Guard
  If you perform your upgrade using Oracle Data Guard to carry out a rolling upgrade, then you must move the Data Guard broker configuration files before starting your upgrade.
- Preparing the New Oracle Home for Upgrading
  After backing up the database that you want to upgrade, prepare the new Oracle home in a new location. Install the software for the new Oracle Database release into the new location.
- Prerequisites for Preparing Oracle Home on Windows
  Your system must meet these requirements before you can upgrade Oracle Database on Microsoft Windows platforms.
- Performing Preupgrade Checks Using AutoUpgrade
  The AutoUpgrade Utility is a Java JAR file provided by Oracle that helps to ensure that your upgrade completes successfully.
- Testing the Upgrade Process for Oracle Database
  Your test plan for Oracle Database upgrades should include these test procedures.

- Requirements for Upgrading Databases That Use Oracle Label Security and Oracle Database Vault
  You must complete these tasks before starting an upgrade with a database using Oracle Label Security or Oracle Database Vault.
- Backing Up Oracle Database for Upgrading
  Use this procedure to back up your existing Oracle Database before you attempt an upgrade.

Tasks to Prepare for Oracle Database Upgrades

Carry out these tasks to prepare your upgrade.

Before you upgrade your database, Oracle recommends that you review the new features and determine the best upgrade path and method to use, and carry out procedures to prepare your database for upgrade. Oracle strongly recommends that you test the upgrade process and prepare a backup strategy.

- Become Familiar with New Oracle Database Features
  Before you plan the upgrade process, become familiar with the features of the new Oracle Database release.
- Review Deprecated and Desupported Features
  Before you upgrade, check to see if deprecated or desupported features require attention in your upgrade plan.
- Choose an Upgrade Method for Oracle Database
  Oracle offers several methods to upgrade your database, which support the complexities of your enterprise.
- Choose a New Location for Oracle Home when Upgrading
  You must choose a location for Oracle home for the new release of Oracle Database that is separate from the Oracle home of your current release.
- Develop a Test Plan for Upgrading Oracle Database
  Review these topics to understand how to create a series of carefully designed tests to validate all stages of the upgrade process.
- Schema-Only Accounts and Upgrading EXPIRED Password Accounts
  Before starting your upgrade, determine if you want to use password authentication to default Oracle Database accounts where their passwords are in `EXPIRED` status, and their account is in `LOCKED` status
- Back Up Files to Preserve Downgrade and Recovery Options
  To ensure that you can recover from upgrade issues, and downgrade to an earlier release if necessary, Oracle recommends that you implement a backup strategy for your database, and for some specific files.

Review Deprecated and Desupported Features

Before you upgrade, check to see if deprecated or desupported features require attention in your upgrade plan.

Every release, Oracle modifies or removes support for features, views, and parameters, so that Oracle can focus on improving core manageability and functionality of other features in the database. For that reason, as part of your upgrade planning, Oracle recommends that you review the list of features listed as deprecated

or desupported in a new release, and determine if these changes are of concern for your applications.

There are two categories of features scheduled for removal:

- **Deprecated** features are features that are no longer being enhanced, but are still supported for the full life of this release of Oracle Database.
- **Desupported** features are features that are no longer supported by fixing bugs related to that feature. Often, Oracle can choose to remove the code required to use the feature. A deprecated feature can be desupported in the next Oracle Database release.

If you see that a feature is deprecated, then Oracle strongly recommends that you stop using that feature as soon as it is practicable for you to do so. Start planning your migration away from deprecated features at the time that they are deprecated.

Become Familiar with New Oracle Database Features

Before you plan the upgrade process, become familiar with the features of the new Oracle Database release.

Oracle Database New Features Guide is a good starting point for learning the differences between Oracle Database releases. Also, check specific guides in the Oracle Database documentation library to find information about new features for a certain component. For example, see *Oracle Real Application Clusters Administration and Deployment Guide* for changes in Oracle Real Application Clusters.

> **Note:**
>
> Oracle Database training classes are an excellent way to learn how to take full advantage of the features and functions available with Oracle Database. You can find more information here:
>
> `http://education.oracle.com/`

Related Topics

- *Oracle Database New Features Guide*

Choose an Upgrade Method for Oracle Database

Oracle offers several methods to upgrade your database, which support the complexities of your enterprise.

- The AutoUpgrade Utility Method for Upgrading Oracle Database
  The AutoUpgrade utility identifies issues before upgrades, performs pre- and postupgrade actions, deploys upgrades, performs postupgrade actions, and starts the upgraded Oracle Database.
- The Graphical User Interface Method for Upgrading Oracle Database
  Database Upgrade Assistant (DBUA) interactively steps you through the upgrade process and configures the database for the new Oracle Database release.

- The Manual, Command-Line Method for Upgrading Oracle Database
  Manual upgrades give you finer control over the upgrade process.
- The Export/Import Method for Migrating Data When Upgrading Oracle Database
  You can use Oracle Data Pump to carry out data exports and imports.

The AutoUpgrade Utility Method for Upgrading Oracle Database

The AutoUpgrade utility identifies issues before upgrades, performs pre- and postupgrade actions, deploys upgrades, performs postupgrade actions, and starts the upgraded Oracle Database.

The AutoUpgrade utility is designed to automate the upgrade process, both before starting upgrades, during upgrade deployments, and during postupgrade checks and configuration migration. You use AutoUpgrade after you have downloaded binaries for the new Oracle Database release, and set up new release Oracle homes. When you use AutoUpgrade, you can upgrade multiple Oracle Database deployments at the same time, using a single configuration file, customized as needed for each database deployment.

The Graphical User Interface Method for Upgrading Oracle Database

Database Upgrade Assistant (DBUA) interactively steps you through the upgrade process and configures the database for the new Oracle Database release.

The preferred option for upgrading Oracle Database is to use the AutoUpgrade utility. However, you can use DBUA to upgrade multitenant architecture container databases (CDB), and pluggable databases (PDBs).

DBUA starts the Pre-Upgrade Information Tool, which fixes some configuration settings to the values required for the upgrade. For example, the tool can change initialization parameters to values required for the upgrade. The tool also provides you with a list of items that you can fix manually before you continue with the upgrade.

The Manual, Command-Line Method for Upgrading Oracle Database

Manual upgrades give you finer control over the upgrade process.

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

Before the Upgrade

- Analyze the database using the Pre-Upgrade Information Tool.

  The Pre-Upgrade Information Tool is a Java JAR file that is supplied with Oracle Database. When you start the tool, it self-extracts, and then executes SQL scripts that perform diagnostics. The tool then displays information about possible upgrade issues, the reason why these issues are occurring, and the actions that you must take to resolve them. The tool also generates fixup scripts for some issues.
- Prepare the new Oracle home.
- Perform a backup of the database.

Depending on the Oracle Database release that you upgrade, you can be required to perform more preupgrade steps. These steps can include adjusting the parameter

file for the upgrade, removing desupported initialization parameters, or adjusting initialization parameters that can cause upgrade problems.

The Export/Import Method for Migrating Data When Upgrading Oracle Database

You can use Oracle Data Pump to carry out data exports and imports.

Topics:

- The Effects of Export/Import on Upgraded Oracle Databases
  Review this topic to understand the benefits of Export/Import data migration.
- Export/Import Benefits for Migrating Data for Oracle Database
  Migrating data when upgrading Oracle Database using Export/Import provides benefits that can increase performance.
- Time Requirements for Migrating Data with Export/Import
  Understand the time it takes for data migration using Oracle Data Pump.

The Effects of Export/Import on Upgraded Oracle Databases

Review this topic to understand the benefits of Export/Import data migration.

The Export/Import data migration 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 newly upgraded Oracle Database 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 newly upgraded Oracle database.

Most importantly, the Export/Import operation results in a completely new database. Although the current target database ultimately contains a copy of the specified data that you migrated, the upgraded database can perform differently from the original source database. Although Export/Import creates an identical copy of the database, other factors can cause unexpected performance issues. (For example: disk placement of data, and unset tuning parameters).

Export/Import Benefits for Migrating Data for Oracle Database

Migrating data when upgrading Oracle Database using Export/Import provides benefits that can increase performance.

Using Export/Import to migrate data provides 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 that you want to populate with imported data.
- Facilitates side-by-side testing of the old and new releases of Oracle Database because an entirely new database is created.

- Enables the copying of specified database objects or users. Importing only the objects, users, and other items you need is useful for establishing a test environment for the new software on only a subset of the production data. Data Pump Export/Import provides flexible data-subsetting capabilities.
- Serves as a backup archive. You can use a full database export as an archive of the current database.
- Enables you to establish the upgraded database on a different operating system or hardware platform than the platform on which your earlier release database is placed.
- Network-based Data Pump Import enables you to load the new release Oracle Database directly across the network for your earlier release Oracle Database. By using network-based Data Pump import, you are not required to use intervening dump files.

Time Requirements for Migrating Data with Export/Import

Understand the time it takes for data migration using Oracle Data Pump.

Migrating an entire Oracle Database by using Oracle Data Pump using Export/Import can take a long time, especially compared to using DBUA or performing a manual upgrade. If you use Oracle Data Pump to migrate data, then schedule the migration during non-peak hours or make provisions for propagating to the new database any changes that are made to the current database during the upgrade.

Choose a New Location for Oracle Home when Upgrading

You must choose a location for Oracle home for the new release of Oracle Database that is separate from the Oracle home of your current release.

Using separate installation locations enables you to keep your existing Oracle software installed along with the new Oracle software. By using separate installation locations, you can test the upgrade process on a test database before replacing your production environment entirely.

When you upgrade a database, whether the database is a non-CDB or a CDB, a new location is needed to install the new Oracle home.

If you are upgrading a PDB by using an unplug/plug upgrade, then the target CDB into which you plug the PDB is the location for the PDB. There is no need to choose a new location for installing the target Oracle homes, because the target CDB already has its Oracle home.

Develop a Test Plan for Upgrading Oracle Database

Review these topics to understand how to create a series of carefully designed tests to validate all stages of the upgrade process.

Oracle recommends that you perform rigorous tests of your database and applications. When you run and complete tests successfully, you help to ensure that you understand the process of upgrading the production database, so that the upgrade process is predictable and successful. Oracle strongly recommends that you perform as much testing as possible before upgrading a production database. Do not underestimate the importance of a complete and repeatable testing process.

You can choose to perform tests manually, or you can use utilities to assist your tests, such as Oracle Real Application Testing features like Database Replay or SQL Performance Analyzer. In either case, the types of tests that you perform are the same.

Your test plan must include these types of tests:

- Upgrade Testing
  When you upgrade Oracle Database to a new release, Oracle strongly recommends that you create, test, and validate an upgrade plan.
- Minimal Testing
  You can find application startup or invocation problems when you perform minimal testing of applications on a test new release environment.
- Functional Testing After Upgrades
  Perform functional testing of the upgraded Oracle Database after the upgrade is complete.
- High Availability Testing
  To ensure that you can continue to meet your service level agreements, plan to perform High Availability testing on your upgraded Oracle Database system.
- Integration Testing to Ensure Applications are Compatible
  Integration testing for Oracle Database examines the interactions among components of the system.
- Performance Testing an Upgraded Oracle Database
  Plan performance testing comparisons between your earlier release and upgraded Oracle Database.
- Volume and Load Stress Testing for Oracle Database Upgrades
  To perform volume and load stress testing of the entire upgraded Oracle Database under high volume and loads, use Database Replay.
- Test Plan Guidelines for Oracle Database Upgrade Planning
  Perform planned tests on your current database and on the test database that you upgraded to the new Oracle Database release.

Upgrade Testing

When you upgrade Oracle Database to a new release, Oracle strongly recommends that you create, test, and validate an upgrade plan.

Upgrade testing for Oracle Database entails planning and testing the upgrade path from your current Oracle Database software to the new Oracle Database release. Oracle strongly recommends that you plan and test your upgrade, whether you use Oracle Database Upgrade Assistant (DBUA), perform a manual upgrade, or use the AutoUpgrade utility. Planning and testing also applies if you use data migration methods, such as Oracle Data Pump Export/Import, or other data-copying methods. Regardless of the upgrade or data migration method you choose, you must plan, test, and validate changes.

Minimal Testing

You can find application startup or invocation problems when you perform minimal testing of applications on a test new release environment.

Minimal testing for Oracle Database entails moving all or part of an application from the current database to the new database, and running the application without enabling any new database features. It is possible that minimal testing does not reveal problems that appear in an actual production environment. However, minimal testing immediately reveals any application startup or invocation problems.

Functional Testing After Upgrades

Perform functional testing of the upgraded Oracle Database after the upgrade is complete.

Functional testing for Oracle Database is a set of tests in which new and existing features and functions 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.

High Availability Testing

To ensure that you can continue to meet your service level agreements, plan to perform High Availability testing on your upgraded Oracle Database system.

High Availability testing for Oracle Database ensures that the upgraded database system meets these recovery business requirements:

- Recovery Time Objective (RTO)
- Recovery Point Objective (RPO)

Oracle recommends the following test procedures for high availability testing:

- Create node or instance failures during stress testing. Node or instance failures help to evaluate the Oracle RAC recovery capability.
- Test fallback plans and procedures to ensure that you can minimize downtime on upgraded databases.
- Check database performance and stability, and resolve performance problems. Resolving performance problems helps to ensure that the upgrade process runs within the time that you have allocated.

Integration Testing to Ensure Applications are Compatible

Integration testing for Oracle Database examines the interactions among components of the system.

Oracle recommends that you carry out the following tests as part of your integration testing:

- To ensure that Pro*C/C++ applications are compatible with the upgraded database, test Pro*C/C++ application clients with the upgraded Oracle Database
- Test graphical user interfaces.
- Test all applications that interact directly or indirectly with the database. Subtle changes in an upgraded Oracle Database, such as data types, data in the data dictionary (additional rows in the data dictionary, object type changes, and so on) can affect front-end applications, even if those applications are not directly connected to the upgraded Oracle Database instance.

- Test and stress-test any Oracle Net or Oracle Net Services connections between components.

Related Topics

- C++ Applications
- Upgrade Considerations for Oracle Net Services

Performance Testing an Upgraded Oracle Database

Plan performance testing comparisons between your earlier release and upgraded Oracle Database.

Performance testing of the upgraded Oracle Database compares the performance of various SQL statements in the new database with the performance of those same statements in the current database. Before upgrading, analyze the performance profile of applications under your current Oracle Database release. Specifically, analyze and understand the calls that applications make to the database server.

Oracle strongly recommends that you set up a testing system with the same storage, data, and other characteristics as your production system.

- Database Replay and Performance Testing
  Use the Database Replay feature to perform real-world testing of a database upgrade on your production workload before actually upgrading the production database.
- SQL Performance Analyzer
  Use the SQL Performance Analyzer to forecast the impact of system changes on a SQL workload.
- SQL Plan Management
  Review this topic to understand how to carry out SQL plan managements after upgrades to avoid performance regressions.

Database Replay and Performance Testing

Use the Database Replay feature to perform real-world testing of a database upgrade on your production workload before actually upgrading the production database.

The Database Replay feature captures the actual database workload on the production system, and replays it on the test system. Database Replay also provides analysis and reporting to highlight potential problems; for example, errors encountered, divergence in performance, and so forth. In addition, all the regular Enterprise Manager performance monitoring and reporting tools such as Automatic Database Diagnostic Monitor, Automatic Workload Repository (AWR), and Active Session History are available to address any problems.

Note:

You can change the stored procedure logic in the database. However, the stored PL/SQL procedures that implement the application logic must maintain the same interfaces as before the upgrade. If an upgrade affects the stored procedures of an application, replaying the workload may not be possible. Using Database Replay tool with the same interfaces provides you with good diagnostics to see if the new application logic in the server is performing as expected after the upgrade.

See Also:

- *Oracle Database Testing Guide* for information about how to capture and replay workloads
- *Oracle Database Performance Tuning Guide* for more information on the Automatic Workload Repository

SQL Performance Analyzer

Use the SQL Performance Analyzer to forecast the impact of system changes on a SQL workload.

SQL Performance Analyzer enables you to evaluate the effect of an upgrade on your SQL workloads. SQL Performance Analyzer finds possible issues by identifying the SQL statements affected by the upgrade. It then measures the performance divergence of SQL workloads before the upgrade, and after the upgrade. The analysis enables you to assess the overall effect of the upgrade on SQL performance. You can then take measures to avoid any negative outcome from SQL workload changes before they can affect users.

See Also:

Oracle Database Testing Guide for further information, and examples of using the SQL Performance Analyzer to perform analysis on potential database changes

SQL Plan Management

Review this topic to understand how to carry out SQL plan managements after upgrades to avoid performance regressions.

A database upgrade that installs a new optimizer version usually results in plan changes for a small percentage of SQL statements. Most plan changes result in no performance change or improvement. However, certain plan changes may cause performance regressions. SQL plan management prevents performance regressions

resulting from sudden changes to the execution plan of a SQL statement by providing components for capturing, selecting, and evolving SQL plan information. SQL plan management is a preventative mechanism that records and evaluates the execution plans of SQL statements over time, and builds SQL plan baselines composed of a set of existing plans that are proven efficient after repeated use. SQL plan management uses the SQL plan baselines to preserve the performance of corresponding SQL statements, regardless of changes occurring in the system.

With SQL plan management, the optimizer automatically manages execution plans and ensures that only known or verified plans are used. When SQL Plan management finds a new plan for a SQL statement, it does not use this plan until the database verifies that the new plan has comparable or better performance than the current plan. If you seed SQL plan management with your current execution plans, then those plans becomes the SQL plan baseline for each statement. The optimizer uses these plans after the upgrade. If the Oracle Database 12*c* optimizer determines that a different plan can result in better performance, then the new plan is queued for verification and is not used until it has been confirmed to have comparable or better performance than the current plan.

There are several ways to seed or populate a SQL Management Base (SMB) with execution plans:

Bulk Load a SQL Management Base from the Cursor Cache

Bulk loading of execution plans or SQL plan baselines from the cursor cache is useful when upgrading a database from Oracle Database 11*g* to the latest release of Oracle Database. The cursor cache is a shared SQL area, and SQL plans that are bulk loaded are automatically accepted and added to existing or new plan histories as SQL plan baselines.

1. In the source release of Oracle Database, use the `DBMS_SPM.LOAD_PLAN_FROM_CURSOR_CACHE` procedure or Oracle Enterprise Manager to load all of the execution plans in the cursor cache into the SQL Management Base.
2. Upgrade the database.

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* for more information on how to load plans from the shared SQL area using PL/SQL or Oracle Enterprise Manager

Bulk Load a SQL Management Base with a SQL Tuning Set (STS)

Bulk loading of execution plans or SQL plan baselines may be done with a SQL Tuning Set. This is useful when upgrading from Oracle Database 10*g*, where no SQL Management Base (SMB) exists to directly load from the cursor cache, or to load historic plans from the Automatic Workload Repository.

1. In the source release of Oracle Database, create an STS that includes the execution plan for each of the SQL statements.
2. Load the STS into a staging table and export the staging table into a dump file.
3. Import the staging table from a dump file into the new release of Oracle and unload the STS.

4. Use Oracle Enterprise Manager or `DBMS_SPM.LOAD_PLANS_FROM_SQLSET` to load the execution plans into the SQL Management Base.

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* for the complete procedure for bulk loading execution plans or SQL plan baselines

Unpack Existing SQL Plan Baselines from a Staging Table

You can test and tune all of your critical SQL queries on an Oracle Database test environment and then move those SQL execution plans to your Oracle Database production environment. Alternatively, you can take plans for SQL queries from your pre-upgrade Oracle Database production environment and move them to your post-upgrade production environment.

1. On the Oracle Database 12*c* test system, after completing all testing and tuning, use the `DBMS_SPM.LOAD_PLAN_FROM_CURSOR_CACHE` procedure or Enterprise Manager to load all of the execution plans in the cursor cache into the SQL Management Base.
2. Create a staging table using the `DBMS_SPM.CREATE_STGTAB_BASELINE` procedure.
3. Pack the SQL plan baselines you created in step 1 into the staging table using the `DBMS_SPM.PACK_STGTAB_BASELINE` function.
4. Export the staging table into a flat file using Data Pump.
5. Transfer this flat file to the target system.
6. Import the staging table from the flat file using Data Pump.
7. Unpack the SQL plan baselines from the staging table into the SQL Management Base on the target system using the `DBMS_SPM.UNPACK_STGTAB_BASELINE` function.

> **See Also:**
>
> - *Oracle Database SQL Tuning Guide* for more information about loading plans from a staging table
> - *Oracle Database Utilities* for information about using Data Pump

Volume and Load Stress Testing for Oracle Database Upgrades

To perform volume and load stress testing of the entire upgraded Oracle Database under high volume and loads, use Database Replay.

Oracle Replay can assist you to uncover load issues before you move an upgraded Oracle Database release to production. **Volume** describes the amount of data being manipulated. **Load** describes the level of concurrent demand on the system. So when you capture and replay a real production system volume and load, you can emulate that load on your upgraded Oracle Database, and observe how it performs under various volumes and loads.

Volume and load stress testing is crucial. However, it is commonly overlooked. After upgrades, Oracle has found that some customers do not conduct any kind of volume or load stress testing. Instead, customers often rely on benchmarks that do not characterize business applications. Benchmarks are valuable: Oracle recommends that you conduct benchmarks of your applications. Benchmarking can help you to uncover problems relating to functions, performance, and integration. However, using benchmarks cannot replace volume and load stress testing.

Load testing involves running an application load against the new Oracle Database release, using an environment with the same data and infrastructure. When you run a load test, you are ensuring that your applications do not encounter problems, such as new errors, or performance issues under the load conditions that you think are likely to occur during production. Many times, problems manifest only under certain load conditions, and are normally not seen in functional testing. The Database Replay feature is ideal for such load testing. Database Replay enables you to capture the system workload from a production environment, and replay it in identical fashion on the test system.

Related Topics

- Introduction to Database Replay

Test Plan Guidelines for Oracle Database Upgrade Planning

Perform planned tests on your current database and on the test database that you upgraded to the new Oracle Database release.

- Compare the test results, noting anomalies.
- Repeat the test upgrade as many times as necessary until issues are resolved.

Test the newly upgraded test database with existing applications to verify that they operate properly with a new Oracle database.

- Test enhanced functions and new capabilities by adding available Oracle Database features.
- Ensure that the applications operate in the same manner as they did in the current database.

> **See Also:**
>
> *Oracle Database Testing Guide* for information on testing a database upgrade

Schema-Only Accounts and Upgrading EXPIRED Password Accounts

Before starting your upgrade, determine if you want to use password authentication to default Oracle Database accounts where their passwords are in `EXPIRED` status, and their account is in `LOCKED` status

During upgrades to Oracle Database 19c and later releases, default Oracle accounts that have not had their passwords reset before upgrade (and are set to `EXPIRED` status), and that are also set to `LOCKED` status, are set to `NO AUTHENTICATION` after the upgrade is complete.

Because of this new feature, default accounts that are changed to schema-only accounts become unavailable for password authentication. The benefit of this feature is that administrators no longer have to periodically rotate the passwords for these Oracle Database-provided schemas. This feature also reduces the security risk of attackers using default passwords to hack into these accounts.

If you want to prevent these Oracle accounts from being set to schema-only accounts during the upgrade, then you must either set a valid strong password for the account before you start the upgrade, or set a valid strong password for these accounts after upgrade, or unlock the accounts before you log in to the upgraded Oracle Database.

After the upgrade, an administrator can also enable password authentication for schema-only accounts. However, for better security, Oracle recommends that you keep these accounts as schema only accounts.

Related Topics

- *Oracle Database Security Guide*

Back Up Files to Preserve Downgrade and Recovery Options

To ensure that you can recover from upgrade issues, and downgrade to an earlier release if necessary, Oracle recommends that you implement a backup strategy for your database, and for some specific files.

- Prepare a Backup Strategy Before Upgrading Oracle Database
  You must design and carry out an appropriate backup strategy to ensure a successful upgrade.
- Oracle Data Guard Broker Configuration File and Downgrades
  With upgrades to Oracle Database 19c and later releases, you must back up the Data Guard broker configuration file to preserve the capability to downgrade to an earlier release.
- Exporting a Broker Configuration
  Use the `EXPORT CONFIGURATION` command to export the metadata contained in the broker configuration file to a text file.

Prepare a Backup Strategy Before Upgrading Oracle Database

You must design and carry out an appropriate backup strategy to ensure a successful upgrade.

For Oracle Database Enterprise Edition, the primary fallback mechanism is Flashback Database. However, you should also have a complete backup strategy in place.

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 is necessary to meet your availability requirements?
- Are backups archived in a safe, offsite location?
- Are backups tested to ensure that they are done properly?
- How quickly can backups be restored (including backups in offsite storage)?
- Have disaster 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. For information about implementing backup strategies using RMAN, review *Oracle Database Backup and Recovery User's Guide*.

Related Topics

- Backing Up the Database
- Using Flashback Database and Restore Points

Oracle Data Guard Broker Configuration File and Downgrades

With upgrades to Oracle Database 19c and later releases, you must back up the Data Guard broker configuration file to preserve the capability to downgrade to an earlier release.

In releases before Oracle Database 19c, Oracle Database settings that are mapped to Oracle Data Guard broker properties are maintained in the Oracle Data Guard broker configuration file, and can be modified using the `DGMGRL` command-line interface. However, starting with Oracle Database 19c, these database settings are no longer stored in the broker configuration file. As a result of this change, although you can continue to modify these properties using DGMGRL, the values that you modify are no longer stored in the Oracle Data Guard broker configuration file. Instead, the DGMGRL commands directly modify the Oracle Database initialization parameters or database settings to which these Oracle Data Guard Broker properties are mapped.

Because of this change to the way that property settings are managed, if you use Oracle Data Guard broker, then Oracle recommends that you export your earlier release Oracle Data Guard broker configuration file to a secure backup location before you start the upgrade. If you do not back up the Oracle Data Guard broker configuration file before the upgrade, then after the upgrade, you cannot downgrade to an earlier release and retain the property options you previously selected for Oracle Data Guard.

Exporting a Broker Configuration

Use the `EXPORT CONFIGURATION` command to export the metadata contained in the broker configuration file to a text file.

The directory in which the broker configuration file is stored must be accessible to the Oracle server process.

1. Connect to the primary database.

```
DGMGRL> CONNECT sysdg@North_Sales.example.com;
Password: password
Connected to "North_Sales"
Connected as SYSDG.
```

2. Export the broker configuration.

   The following command exports the broker configuration and stores it in a file named `myconfig.txt` in the trace directory.

```
DGMGRL> EXPORT CONFIGURATION TO 'myconfig.txt';
Succeeded.
```

Pre-Upgrade Information Tool Checklist

To obtain a checklist of tasks you must complete before upgrading an Oracle Database on a physical server or virtual machine, run the Pre-Upgrade Information Tool (`preupgrade.jar`).

Before starting your upgrade, ensure that you have a new release Oracle Database installed and configured that you can use as the target for your upgrade. When your target Oracle Database home is prepared, then run the Pre-Upgrade Information Tool (`preupgrade.jar`) on your system, using the instructions that you can find in this guide.

Oracle requires that you run the Pre-Upgrade Information Tool before you upgrade Oracle Database. The tool can identify issues for you to address before you start your upgrade. In certain cases, the tool can also resolve some of them.

> **Tip:**
>
> Consider reviewing Oracle's upgrade blog for tips and suggestions that can assist you with your upgrade preparations.

Related Topics

- Upgrade your Database – NOW! Mike Dietrich's Oracle Database Upgrade Blog
- *Oracle Database PL/SQL Packages and Types Reference*

Installing the New Oracle Database Software for Single Instance

Use this procedure overview to assist you to install the software for the new Oracle Database release for a single instance deployment.

> **Note:**
>
> You cannot upgrade a database using Database Upgrade Assistant (DBUA) when the source and target Oracle homes are owned by different users. Attempting to do so returns error `PRKH-1014`. Either ensure that the source and target databases have the same owner, or perform a manual upgrade.

To install the new Oracle Database software for this release:

1. Follow the instructions in your Oracle operating system-specific documentation to prepare for installation of Oracle Database software.
2. Start Oracle Universal Installer, and select a software-only installation.

   When installation of Oracle Database software has completed successfully, click **Exit** to close Oracle Universal Installer.

3. Run the Pre-Upgrade Information Tool. The tool enables you to check the types of items that DBUA checks. The tool identifies issues, and can help you fix some issues that it finds.

   By default, the tool is in the location *New_release_Oracle_home*`/rdbms/admin/preupgrade.jar`.

   If you use Oracle Label Security, Oracle Database Vault, or both, then select **Enterprise Edition** on the Select Database Edition page, click **Select Options**, and enable one or both components from the components list.

Installing the New Oracle Database Software for Oracle RAC

Use this procedure overview to assist you to install the software for the new Oracle Database release for an Oracle RAC deployment.

> **Note:**
>
> You cannot upgrade a database using Database Upgrade Assistant (DBUA) when the source and target Oracle homes are owned by different users. Attempting to do so returns error PRKH-1014. Either ensure that the source and target databases have the same owner, or perform a manual upgrade.

If you are upgrading an Oracle RAC database, then you must perform the following steps in the order shown:

1. Upgrade Oracle Clusterware:
   a. Upgrade Oracle Clusterware first as described in the Oracle Grid Infrastructure installation guide for your operating system.
   b. Mount the Oracle Grid Infrastructure installation media.
   c. Perform operating system prerequisite checks on each of the nodes that you intend to upgrade, to ensure that they meet the system prerequisites for Oracle Grid Infrastructure (Oracle Clusterware and Oracle ASM).
   d. If necessary, perform patch upgrades of the earlier release of Oracle Clusterware software to the most recent patch version.
   e. Ensure that you are logged in as the user that owns the Oracle Grid Infrastructure installation, and run the Oracle Grid Infrastructure installation. Provide information as prompted by the installer.
   f. When prompted, open a separate terminal session, log in as `root`, and run `root.sh`.
2. After upgrading Oracle Clusterware, follow the instructions in your Oracle operating system-specific documentation to prepare for Oracle Database software installation.
3. Run the Pre-Upgrade Information Tool. You can preview the types of items DBUA checks, and identify issues. The Pre-Upgrade Information Tool can help you fix some of the prerequisite issues that it finds.

4. Start Oracle Universal Installer, and install the software.

   When installation of Oracle Database software has completed successfully, click **Exit** to close Oracle Universal Installer.

5. Run DBUA.

   If you use Oracle Label Security, Oracle Database Vault, or both, then select **Enterprise Edition** on the Select Database Edition page, click **Select Options**, and enable one or both components from the components list.

Related Topics

- *Oracle Clusterware Installation Guide* for your platform

Database Preparation Tasks to Complete Before Starting Oracle Database Upgrades

Ensure that you have completed these database preparation tasks before starting an Oracle Database upgrade.

- Release Updates and Requirements for Upgrading Oracle Database
  Before starting upgrades, update your new release Oracle home to the latest Release Update (Update).
- Copying Transparent Encryption Oracle Wallets
  If you use Oracle wallet with Transparent Data Encryption (TDE), then ensure that the new release `sqlnet.ora` has a proper configuration for the wallet location (`network/admin`) folder.
- Recommendations for Oracle Net Services When Upgrading Oracle Database
  You must ensure that the listener is running in your new release Oracle home.
- Understanding Password Case Sensitivity and Upgrades
  By default, Oracle Database 12*c* Release 2 (12.2) and later releases are upgraded to an Exclusive Mode. Exclusive Modes do not support case-insensitive password-based authentication.
- Checking for Accounts Using Case-Insensitive Password Version
  Use these procedures to identify if the Oracle Database that you want to upgrade has accounts or configuration parameters that are using a case-insensitive password version.
- Check for Profile Scripts (glogin.sql and login.sql)
  For all upgrade methods, Oracle recommends that you run upgrades without the use of profile scripts.
- Running Upgrades with Read-Only Tablespaces
  Use the Parallel Upgrade Utility with the `-T` option to take schema-based tablespaces offline during upgrade.
- High Availability Options for Oracle Database
  Review the high availability options available to you for Oracle Database using Standard Edition High Availability, Oracle Restart, Oracle Real Application Clusters (Oracle RAC), and Oracle RAC One Node.
- Options for High Availability with Oracle Database Standard Edition
  To enable high availability for Oracle Database Standard Edition in releases after Oracle Database 19c, learn how you can use Standard Edition High Availability.

- Non-CDB Upgrades and Oracle GoldenGate
  If you are upgrading a Non-CDB Oracle Database where Oracle GoldenGate is deployed, then you must shut down Oracle GoldenGate, and reconfigure it after conversion and upgrade for the multitenant architecture.

Release Updates and Requirements for Upgrading Oracle Database

Before starting upgrades, update your new release Oracle home to the latest Release Update (Update).

The software for new Oracle Database releases contains a full release that includes all the latest updates for Oracle Database at the time of the release.

Before you start an upgrade, Oracle strongly recommends that you update your new release Oracle home to the latest quarterly Release Update (Update).

My Oracle Support provides detailed notes about how you can obtain the updates, as well as tools for lifecycle management.. For example:

- My Oracle Support note 2118136.2 contains a download assistant to help you select the updates, revisions, Patch Set Updates (PSU), SPU (CPU), Bundle Patches, Patchsets, and Base Releases that you need for your environment. Oracle highly recommends that you start here.
- My Oracle Support note 1227443.1 contains a list of Oracle Database PSU/BP/Update/Revision known issues. This note provides information about all known issues notes for Oracle Database, Oracle Grid Infrastructure, and the Oracle JavaVM Component (OJVM).

Related Topics

- My Oracle Support Note 2118136.2
- My Oracle Support Note 1227443.1

Copying Transparent Encryption Oracle Wallets

If you use Oracle wallet with Transparent Data Encryption (TDE), then ensure that the new release `sqlnet.ora` has a proper configuration for the wallet location (`network/admin`) folder.

> **Note:**
>
> This procedure using `sqlnet.ora` to configure keystores is deprecated. Oracle recommends that you use the instance initialization file-based approach using the `WALLET_ROOT` and `TDE_CONFIGURATION` initialization parameters. After the upgrade, review and configure the `WALLET_ROOT` initialization parameter.

If you use the `sqlnet.ora` file to configure keystores, then you must copy `sqlnet.ora` and the wallet file manually to a keystore location outside of the Oracle home. Wallets should be stored in a location outside of an Oracle Home. If you move the wallet, then you must update `sqlnet.ora` before starting the upgrade. For auto-login wallets, you must also copy the `cwallet.sso` file manually to the new keystore location.

1. Log in as an authorized user.
2. Manually copy the `sqlnet.ora` file, and the wallet file, `ewallet.p12`, to the new release Oracle home.
3. If you have enabled an auto-login wallet, then also copy the `cwallet.sso` file to the new release Oracle home. There is no need to complete the next step. If you have not enabled an auto-login wallet, then proceed to the next step.
4. If you have not enabled an auto-login wallet, then open the Oracle wallet in mount.

   For example:

   ```
   SQL> STARTUP MOUNT;
   SQL> ALTER SYSTEM SET ENCRYPTION WALLET OPEN
   ```

   To create a local auto-login keystore for a software keystore, use the following syntax:

   ```
   ADMINISTER KEY MANAGEMENT CREATE LOCAL AUTO_LOGIN KEYSTORE
   FROM KEYSTORE 'keystore_location'
   IDENTIFIED BY software_keystore_password;
   ```

   In this specification:

   - `LOCAL` enables you to create a local auto-login software keystore. Otherwise, omit this clause if you want the keystore to be accessible by other computers.
   - *`keystore_location`* is the path to the keystore directory location of the keystore that is configured in the `sqlnet.ora` file.
   - *`software_keystore_password`* is the existing password of the configured software keystore.

Related Topics

- About the Keystore Location in the sqlnet.ora File

Recommendations for Oracle Net Services When Upgrading Oracle Database

You must ensure that the listener is running in your new release Oracle home.

If the Oracle Database that you are upgrading does not have a listener configured, then before you start the upgrade, you must run Oracle Net Configuration Assistant (`NETCA`) to configure the listening protocol address and service information for the new release of Oracle Database, including a `listener.ora` file. The current listener is backward-compatible with earlier Oracle Database releases.

If you are upgrading Oracle Real Application Clusters Oracle Database, or a release older than Oracle Database 12c, then review the following additional information.

When you upgrade an Oracle RAC database with DBUA, it automatically migrates the listener from your old Oracle home to the new Oracle Grid Infrastructure home. You must administer the listener by using the `lsnrctl` command in the Oracle Grid Infrastructure home. Do not attempt to use the `lsnrctl` commands from Oracle home locations for earlier releases.

In Oracle Database, underlying net services parameters enable data compression, which reduces the size of the session data unit that is transmitted over a SQL TCP connection.

The following new parameters for the `sqlnet.ora` file specify compression, and the preferred compression scheme:

- `SQLNET.COMPRESSION`
- `SQLNET.COMPRESSION_LEVELS`
- `SQLNET.COMPRESSION_THRESHOLD`

These parameters, which were introduced with Oracle Database 12c, are not supported in earlier releases.

Related Topics

- *Oracle Database Net Services Reference*

Understanding Password Case Sensitivity and Upgrades

By default, Oracle Database 12*c* Release 2 (12.2) and later releases are upgraded to an Exclusive Mode. Exclusive Modes do not support case-insensitive password-based authentication.

Accounts that have only the `10G` password version become inaccessible when the server runs in an Exclusive Mode.

In previous Oracle Database releases, you can configure the authentication protocol so that it allows case-insensitive password-based authentication by setting `SEC_CASE_SENSITIVE_LOGON=FALSE`. Starting with Oracle Database 12*c* release 2 (12.2), the default password-based authentication protocol configuration excludes the use of the case-insensitive `10G` password version. By default, the `SQLNET.ORA` parameter `SQLNET.ALLOWED_LOGON_VERSION_SERVER` is set to `12`, which is an Exclusive Mode. When the database is configured in Exclusive Mode, the password-based authentication protocol requires that one of the case-sensitive password versions (`11G` or `12C`) is present for the account being authenticated. This mode excludes the use of the `10G` password version used in earlier releases. After upgrading to Oracle Database 12*c* release 2 and later releases, accounts that have only the case-insensitive `10G` password version become inaccessible. This change occurs because the server runs in an Exclusive Mode by default. When Oracle Database is configured in Exclusive Mode, it cannot use the old `10G` password version to authenticate the client. The server is left with no password version with which to authenticate the client.

For greater security, Oracle recommends that you leave case-sensitive password-based authentication enabled. This setting is the default. However, you can temporarily disable case-sensitive authentication during the upgrade to new Oracle Database releases. After the upgrade, you can then decide if you want to enable the case-sensitive password-based authentication feature as part of your implementation plan to manage your password versions.

Before upgrading, Oracle recommends that you determine if this change to the default password-based authentication protocol configuration affects you. Perform the following checks:

- Identify if you have accounts that use only `10G` case-insensitive password authentication versions.

- Identify if you have Oracle Database 11*g* release 2 (11.2.0.3) database or earlier clients that have not applied critical patch update `CPUOct2012`, or a later patch update, and have any account that does not have the case-insensitive `10G` password version.
- Ensure that you do not have the deprecated parameter SEC_CASE_SENSITIVE_LOGON set to FALSE. Setting this parameter to FALSE prevents the use of the case-sensitive password versions (the `11G` and `12C` password versions) for authentication.

Options for Accounts Using Case-Insensitive Versions

If you have user accounts that have only the case-insensitive `10G` password version, then you must choose one of the following alternatives:

- Before upgrade, update the password versions for each account that has only the `10G` password version. You can update the password versions by expiring user passwords using the `10G` password version, and requesting that these users log in to their account. When they attempt to log in, the server automatically updates the list of password versions, which includes the case-sensitive password versions.
- Change the setting of the SQLNET.ORA parameter `SQLNET.ALLOWED_LOGON_VERSION_SERVER` to any of the settings that are not Exclusive Mode. For example: `SQLNET.ALLOWED_LOGON_VERSION_SERVER=11`

Related Topics

- *Oracle Database 2 Day DBA*
- *Oracle Database Net Services Reference*
- *Oracle Database Security Guide*

Checking for Accounts Using Case-Insensitive Password Version

Use these procedures to identify if the Oracle Database that you want to upgrade has accounts or configuration parameters that are using a case-insensitive password version.

By default, in Oracle Database 12*c* release 2 (12.2) and later releases, the `10G` password version is not generated or allowed.

If you do not set `SQLNET.ALLOWED_LOGON_VERSION_SERVER` to a permissive authentication protocol that permits case-insensitive versions, and you do not want user accounts authenticated with case-insensitive password versions to be locked out of the database, then you must identify affected accounts, and ensure that they are using case-sensitive password versions.

Example 2-1 Finding User Accounts That Use Case-Insensitive (10G) Version

Log in to SQL*Plus as an administrative user, and enter the following SQL query:

```
SELECT USERNAME,PASSWORD_VERSIONS FROM DBA_USERS;
```

The following result shows password versions for the accounts:

```
USERNAME                       PASSWORD_VERSIONS
------------------------------ -----------------
```

```
JONES                          10G 11G 12C
ADAMS                          10G 11G
CLARK                          10G 11G
PRESTON                        11G
BLAKE                          10G
```

In this example, the backgrounds for each user account password verification version in use are different:

- JONES was created in Oracle Database 10G, and the password for JONES was reset in Oracle Database 12C when the setting for the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter was set to 8. As a result, this password reset created all three versions. 11G and 12C use case-sensitive passwords.
- ADAMS and CLARK were originally created with the 10G version, and then 11G, after they were imported from an earlier release. These account passwords were then reset in 11G, with the deprecated parameter SEC_CASE_SENSITIVE_LOGON set to TRUE.
- The password for BLAKE was created with the 10G version, and the password has not been reset. As a result, user BLAKE continues to use the 10G password version, which uses a case-insensitive password.

The user BLAKE has only the 10G password version before upgrade:

```
SQL> SELECT USERNAME,PASSWORD_VERSIONS FROM DBA_USERS;

USERNAME PASSWORD_VERSIONS
------------------------------ ----------------
BLAKE 10G
```

If you upgrade to a new Oracle Database release without taking any further action, then this account becomes inaccessible. Ensure that the system is not configured in Exclusive Mode (by setting the SQLNET.ORA parameter SQLNET.ALLOWED_LOGON_VERSION_SERVER to a more permissive authentication mode) before the upgrade.

Example 2-2 Fixing Accounts with Case-Insensitive Passwords

Complete the following procedure:

1. Use the following SQL query to find the accounts that only have the 10G password version:

   ```
   select USERNAME
      from DBA_USERS
     where ( PASSWORD_VERSIONS = '10G '
           or PASSWORD_VERSIONS = '10G HTTP ')
       and USERNAME <> 'ANONYMOUS';
   ```

2. Configure the system so that it is not running in Exclusive Mode by editing the setting of the SQLNET.ORA parameter SQLNET.ALLOWED_LOGON_VERSION_SERVER to a level appropriate for affected accounts. For example:

   ```
   SQLNET.ALLOWED_LOGON_VERSION_SERVER=11
   ```

   After you make this change, proceed with the upgrade.

3. After the upgrade completes, use the following command syntax to expire the accounts you found in step 1, where *username* is the name of a user returned from the query in step 1:

   ```
   ALTER USER username PASSWORD EXPIRE;
   ```

4. Ask the users for whom you have expired the passwords to log in.
5. When these users log in, they are prompted to reset their passwords. The system internally generates the missing `11G` and `12C` password versions for their account, in addition to the `10G` password version. The `10G` password version is still present, because the system is running in the permissive mode.
6. Ensure that the client software with which users are connecting has the `O5L_NP` capability flag.

   > **Note:**
   >
   > All Oracle Database release 11.2.0.4 and later clients, and all Oracle Database release 12.1 and later clients have the `O5L_NP` capability. Other clients require the `CPUOct2012` patch to acquire the `O5L_NP` capability.
   >
   > The `O5L_NP` capability flag is documented in *Oracle Database Net Services Reference*, in the section on the parameter `SQLNET.ALLOWED_LOGON_VERSION_SERVER`.

7. After all clients have the `O5L_NP` capability, raise the server security back to Exclusive Mode by using the following procedure:

   a. Remove the `SEC_CASE_SENSITIVE_LOGON` setting from the instance initialization file, or set the `SEC_CASE_SENSITIVE_LOGON` instance initialization parameter to `TRUE`. For example:

      ```
      SEC_CASE_SENSITIVE_LOGON = TRUE
      ```

   b. Remove the `SQLNET.ALLOWED_LOGON_VERSION_SERVER` parameter from the server `SQLNET.ORA` file, or set it back to Exclusive Mode by changing the value of `SQLNET.ALLOWED_LOGON_VERSION_SERVER` in the server `SQLNET.ORA` file back to `12`. For example:

      ```
      SQLNET.ALLOWED_LOGON_VERSION_SERVER = 12
      ```

8. Use the following SQL query to find the accounts that still have the `10G` password version:

   ```
   select USERNAME
     from DBA_USERS
    where PASSWORD_VERSIONS like '%10G%'
      and USERNAME <> 'ANONYMOUS';
   ```

9. Use the list of accounts returned from the query in step 8 to expire all the accounts that still have the `10G` password version. Expire the accounts using the following syntax, where *username* is a name on the list returned by the query:

   ```
   ALTER USER username PASSWORD EXPIRE;
   ```

10. Request the users whose accounts you expired to log in to their accounts.

    When the users log in, they are prompted to reset their password. The system internally generates only the `11G` and `12C` password versions for their account.

Because the system is running in Exclusive Mode, the 10G password version is no longer generated.

11. Check that the system is running in a secure mode by rerunning the query from step 1. Ensure that no users are found. When the query finds no users, this result means that no 10G password version remains present in the system.

Example 2-3 Checking for the Presence of SEC_CASE_SENSITIVE_LOGON Set to FALSE

Oracle Database does not prevent the use of the FALSE setting for SEC_CASE_SENSITIVE_LOGON when the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter is set to 12 or 12a. This setting can result in all accounts in the upgraded database becoming inaccessible.

```
SQL> SHOW PARAMETER SEC_CASE_SENSITIVE_LOGON

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
sec_case_sensitive_logon             boolean     FALSE
```

You can change this parameter by using the following command:

```
SQL> ALTER SYSTEM SET SEC_CASE_SENSITIVE_LOGON = TRUE;

System altered.
```

> **Note:**
>
> Unless the value for the parameter SQLNET.ALLOWED_LOGON_VERSION_SERVER is changed to a version that is more permissive than 12, such as 11, do not set the SEC_CASE_SENSITIVE_LOGON parameter to FALSE.

Related Topics

- *Oracle Database Net Services Reference*
- *Oracle Database Security Guide*

Check for Profile Scripts (glogin.sql and login.sql)

For all upgrade methods, Oracle recommends that you run upgrades without the use of profile scripts.

Depending on the content of profile scripts (glogin.sql and login.sql), there is a risk that these scripts can interfere with the upgrade of Oracle Database, and that you can encounter an UPG-1400 UPGRADE FAILED error, or Unexpected error encountered in catcon, or ORA-04023: Object SYS.STANDARD could not be validated or authorized. Oracle recommends that you remove the site profile script (glogin.sql) from the target Oracle home (located in the Oracle home under /sqlplus/admin) before starting the upgrade. Also ensure that no user profile script is defined, either in the current directory, or specified using the environment variable SQLPATH.

Related Topics

- Upgrade and Profile Scripts

Running Upgrades with Read-Only Tablespaces

Use the Parallel Upgrade Utility with the `-T` option to take schema-based tablespaces offline during upgrade.

Oracle Database can read file headers created in earlier releases, so you are not required to do anything to them during the upgrade. The file headers of `READ ONLY` tablespaces are updated when they are changed to `READ WRITE`.

If the upgrade suffers a catastrophic error, so that the upgrade is unable to bring the tablespaces back online, then review the upgrade log files. The log files contain the actual SQL statements required to make the tablespaces available. To bring the tablespaces back online, you must run the SQL statements in the log files for the database, or run the log files for each PDB.

Viewing Tablespace Commands in Upgrade Log Files

If a catastrophic upgrade failure occurs, then you can navigate to the log directory (*`Oracle_base`*`/cfgtoologs/dbua`), and run commands in the log files manually to bring up tablespaces. You can view tablespace commands in the following log files:

- Non-CDB Upgrades: `catupgrd0.log`
- PDB databases: `catupgrd`*`pdbname`*`0.log`, where *`pdbname`* is the name of the PDB that you are upgrading.

At the beginning of each log file, you find SQL statements such as the following, which sets tables to `READ ONLY`:

```
SQL> ALTER TABLESPACE ARGROTBLSPA6 READ ONLY;

Tablespace altered.

SQL> ALTER TABLESPACE ARGROTBLSPB6 READ ONLY;

Tablespace altered.
```

Near the end of each log file, you find SQL statements to reset tables to `READ WRITE`:

```
SQL> ALTER TABLESPACE ARGROTBLSPA6 READ WRITE;

Tablespace altered.

SQL> ALTER TABLESPACE ARGROTBLSPB6 READ WRITE;

Tablespace altered.
```

> **See Also:**
>
> *Oracle Database Administrator's Guide* for information about transporting tablespaces between databases

High Availability Options for Oracle Database

Review the high availability options available to you for Oracle Database using Standard Edition High Availability, Oracle Restart, Oracle Real Application Clusters (Oracle RAC), and Oracle RAC One Node.

The following is an overview of the high availability options available to you for Oracle Database.

Standard Edition High Availability

- Cluster-based active/passive Oracle Database failover solution
- Designed for single instance Standard Edition Oracle Databases
- Available with Oracle Database 19c release update (RU) 19.7 and later
- Requires Oracle Grid Infrastructure 19c RU 19.7 and later, installed as a Standalone Cluster

Oracle Restart

- Oracle Database instance restart only feature and basis for Oracle Automatic Storage Management (Oracle ASM) for standalone server deployments
- For single instance Oracle Databases
- Requires Oracle Grid Infrastructure for a standalone server (no cluster)

Oracle Real Application Clusters (Oracle RAC) One Node

- Provides a cluster-based active/passive Oracle Database failover and online database relocation solution
- Available for Oracle RAC-enabled Oracle Databases
- Only one instance of an Oracle RAC-enabled Oracle Database is running under normal operations
- Enables relocation of the active instance to another server in the cluster in an online fashion. To relocate the active instance, you can temporarily start a second instance on the destination server, and relocate the workload
- Supports Rolling Upgrades - patch set, database, and operating system
- Supports Application Continuity
- Requires Oracle Grid Infrastructure to be installed as a Standalone Cluster

Oracle Real Application Clusters (Oracle RAC)

- Provides active / active Oracle Database high availability and scalability solution
- Enables multiple servers to perform concurrent transactions on the same Oracle Database

- Provides high availability: a failure of a database instance or server does not interrupt the database service as a whole, because other instances and their servers remain operational
- Supports Rolling Upgrades - patch set, database, and operating system
- Supports Application Continuity
- Requires Oracle Grid Infrastructure to be installed as a Standalone Cluster

Options for High Availability with Oracle Database Standard Edition

To enable high availability for Oracle Database Standard Edition in releases after Oracle Database 19c, learn how you can use Standard Edition High Availability.

- Preparing to Upgrade Standard Edition Oracle RAC or Oracle RAC One Node
  To maintain high availability after migrating from Standard Edition Oracle Real Application Clusters (Oracle RAC), you can use Standard Edition High Availability.
- Requirements for Using Standard Edition High Availability With Oracle Databases
  To use Standard Edition High Availability, deploy Oracle Database Standard Edition 2 in accordance with these configuration requirements.

Preparing to Upgrade Standard Edition Oracle RAC or Oracle RAC One Node

To maintain high availability after migrating from Standard Edition Oracle Real Application Clusters (Oracle RAC), you can use Standard Edition High Availability.

Starting with the Oracle Database 19c release, Oracle Database Standard Edition 2 does not support Oracle RAC. To continue to meet high availability needs for Oracle Database Standard Edition, Oracle is introducing Standard Edition High Availability.

Requirements for Using Standard Edition High Availability With Oracle Databases

To use Standard Edition High Availability, deploy Oracle Database Standard Edition 2 in accordance with these configuration requirements.

- The database is created in a cluster running Oracle Grid Infrastructure for a Standalone Cluster, with its database files placed in Oracle Automatic Storage Management (Oracle ASM) or Oracle Automatic Storage Management Cluster File System (Oracle ACFS).
- When using the Database Configuration Assistant, do not create a listener when creating an Oracle Database Standard Edition 2 database that you want to configure for Standard Edition High Availability.
- Register the database with Single Client Access Name (SCAN) listeners as remote listeners, and node listeners as the local listener.
- Create a database service. Use this service, instead of the default database service, when you connect applications or database clients to the database.
- Ensure that the server parameter file (`spfile`) and password file are on Oracle ASM or Oracle ACFS. If the `spfile` and password file were placed on a local file system when the database was created or configured, then move these files to Oracle ASM or Oracle ACFS.

Refer to the database installation documentation for additional requirements that must be met.

Related Topics

- *Oracle Database Installation Guide for Linux*

Non-CDB Upgrades and Oracle GoldenGate

If you are upgrading a Non-CDB Oracle Database where Oracle GoldenGate is deployed, then you must shut down Oracle GoldenGate, and reconfigure it after conversion and upgrade for the multitenant architecture.

If you are using Oracle GoldenGate with the non-CDB Oracle Database that you want to upgrade, then before you convert and upgrade the source non-CDB Oracle Database to the multitenant architecture, you must shut down and remove the Oracle GoldenGate processes, and then reconfigure them after conversion and upgrade for the multitenant architecture. The following is a high level overview of the processes required:

1. Drop Oracle GoldenGate users on the source Oracle Database.
2. Wait until the Oracle GoldenGate processes finish processing all current DML and DDL data in the Oracle GoldenGate trails, and processes are at End of File (EOF).
3. Stop all Oracle GoldenGate processes on the source database.
4. Complete the conversion and upgrade of the source non-CDB Oracle Database to the target Oracle Database on the target release CDB.
5. Restart the database.
6. If you are also upgrading the database from an earlier release to a later major release family (for example, from Oracle Database 12.1 to Oracle Database 19c, which is the terminal patch set of the Oracle Database 12.2 family), then you must install a new version of Oracle GoldenGate that is supported for Oracle Database 19c. If you are upgrading both Oracle Database and Oracle GoldenGate simultaneously, then you must upgrade the database first.

After the database conversion and upgrade is complete, you can create new credentials for the Oracle GoldenGate extract user. With the new credentials you can then create a new Extract process and Extract pump and distribution service for the upgraded Oracle Database PDB on the target CDB, and start up the newly created processes. For more information about completing those procedures after the upgrade, refer to the Oracle GoldenGate documentation.

Related Topics

- Establishing Oracle GoldenGate Credentials
- Configuring Oracle GoldenGate in a Multitenant Container Database

Preparing for Database Rolling Upgrades Using Oracle Data Guard

If you perform your upgrade using Oracle Data Guard to carry out a rolling upgrade, then you must move the Data Guard broker configuration files before starting your upgrade.

The default location for the DB_BROKER_CONFIG files is in the dbs directory in the earlier release Oracle Database Oracle home. When you perform a rolling upgrade of database instances using Oracle Data Guard, you must move the DG_BROKER_CONFIG files to a mount point location outside of the earlier release Oracle home. Also ensure that the DG_BROKER_CONFIG_FILE*n* parameters specify that location, instead of a location in the earlier release Oracle home. During database upgrade, don't migrate the listener. After the upgrade is complete, stop the listener, shut down the database, copy over the listener.ora and tnsnames.ora from the earlier source Oracle Database release environment to the new Oracle Database release environment, and start the listener and database

Tasks Before Starting Your Upgrade

To enable access to the DB_BROKER_CONFIG files during a rolling upgrade, you must complete the following tasks before starting the upgrade

1. Before you start the upgrade, set the Oracle Data Guard files DG_BROKER_CONFIG_FILE1 and DG_BROKER_CONFIG_FILE2 to a separate mount point on your server that is outside of the Oracle home path for either the source or target Oracle Database Oracle homes.
2. Complete a successful upgrade of your earlier release Oracle home to the new Oracle Database release.

Tasks During the Upgrade

Do not migrate the listener during the upgrade.

Oracle recommends that you use AutoUpgrade to complete the upgrade. See:

AutoUpgrade and Oracle Data Guard

Tasks After Completing Your Upgrade

1. Stop the listener for the new release Oracle Database.
2. Shut down the new release Oracle Database.
3. Copy over the listener.ora and tnsnames.ora files from the earlier release Oracle Database to the new release Oracle Database.
4. Start the listener and new release Oracle Database

Refer to *Oracle Data Guard Broker* for information about moving your Data Guard broker configuration files.

Related Topics

- Renaming the Broker Configuration Files

Preparing the New Oracle Home for Upgrading

After backing up the database that you want to upgrade, prepare the new Oracle home in a new location. Install the software for the new Oracle Database release into the new location.

1. (Manual upgrades only) Copy configuration files from the Oracle home of the database being upgraded to the new release Oracle Database Oracle home. If you are using DBUA, then you can ignore this step, because the configuration files are copied for you automatically.

Use the following procedure to copy configuration files to the new Oracle home:

a. If your parameter file resides within the old environment Oracle home, then copy it to the new Oracle home. By default, Oracle looks for the parameter file in the *ORACLE_HOME*/dbs directory on Linux or UNIX platforms and in the *ORACLE_HOME*\database directory on Windows operating systems. After upgrade, the parameter file can reside anywhere else, but it cannot reside in the Oracle home of the old environment.

> **Note:**
>
> If necessary, create a text initialization parameter file (PFILE) from the server parameter file (SPFILE) so that you can edit the initialization parameters.

b. If your parameter file resides within an Oracle ASM instance, then back up the parameter file using the following command:

```
CREATE pfile FROM spfile;
```

If you must downgrade the database and your SPFILE resided within Oracle ASM, then you must restore the parameter file before the downgrade.

c. If your parameter file is a text-based initialization parameter file with either an IFILE (include file) or a SPFILE (server parameter file) entry, and the file specified in the IFILE or SPFILE entry resides within the old environment Oracle home, then copy the file specified by the IFILE or SPFILE entry to the new Oracle home. The file specified in the IFILE or SPFILE entry contains additional initialization parameters.

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

The name and location of the password file are operating system-specific. Where *SID* is your Oracle instance ID, you can find the password file in the following locations:

- Linux or UNIX platforms: The default password file is orapw *SID*. It is located in the directory *ORACLE_HOME*/dbs.
- Windows operating systems: The default password file is pwd*SID*.ora. It is located in the directory *ORACLE_HOME*\database.

2. Adjust your parameter file in the new Oracle Database release by completing the following steps:

a. Remove desupported initialization parameters and adjust deprecated initialization parameters. In new releases, some parameters are desupported, and other parameters are deprecated. Remove all desupported parameters from any parameter file that starts the new Oracle Database instance. Desupported parameters can cause errors in new Oracle Database releases. Also, alter any parameter whose syntax has changed in the new release.

The Pre-Upgrade Information Tool displays any deprecated parameters and desupported parameters it finds in the **Deprecated Parameters** and **Desupported Parameters** sections, respectively.

Adjust the values of the initialization parameters to at least the minimum values indicated by the Pre-Upgrade Information Tool.

Make sure that all path names in the parameter file are fully specified. You should not have relative path names in the parameter file.

b. If the parameter file contains an `IFILE` entry, then change the `IFILE` entry in the parameter file. The `IFILE` entry should point to the new location text initialization parameter file that you specified in step 1. Also edit the file specified in the `IFILE` entry in the same way that you edited the parameter file in step 1.

c. If you are upgrading a cluster database, then if necessary, you can modify the `SPFILE` or `init`*`ORACLE_SID`*`.ora` files.

After making these parameter file adjustments, make sure that you save all of the files that you modified.

3. (Manual upgrades only) If you are upgrading a cluster database, and you are not using Database Upgrade Assistant (DBUA) then you must manually separate the database instance from the cluster. Set the `CLUSTER_DATABASE` initialization parameter to `false`. After the upgrade, you must set this initialization parameter back to `true`. If you are using DBUA, then the assistant takes care of this task for you.

Prerequisites for Preparing Oracle Home on Windows

Your system must meet these requirements before you can upgrade Oracle Database on Microsoft Windows platforms.

For security reasons, different Microsoft Windows user accounts configured as Oracle home users for different Oracle homes are not allowed to share the same Oracle Base.

- Database upgrade is supported when the same Windows user account is used as the Oracle home user in both the *source* and *destination* Oracle homes.
- Database upgrade is supported when the Oracle home from which the database is being upgraded uses the Windows Built-in Account. Releases earlier than Oracle Database 12*c* (release 11.2 and earlier) only supported the built-in account option for the Oracle home user on Windows.
- The Oracle home user may not have access to files outside its own Oracle Base and Oracle home. If that is the case, then if you choose a different Oracle Base during upgrade, it is possible that Oracle Database services cannot access files in the older Oracle Base. Using DBUA for database upgrade ensures that the Oracle home user has access to files outside of its own Oracle Base and its own Oracle home.

  Before upgrading manually, or before using the custom files from the older Oracle Base (for example, wallets, configuration files and other custom files), you must grant access to the Oracle home user for these outside files, or copy these files to the new Oracle Base.

> **See Also:**
>
> *Oracle Database Platform Guide for Microsoft Windows* for information about database administration on Windows

Performing Preupgrade Checks Using AutoUpgrade

The AutoUpgrade Utility is a Java JAR file provided by Oracle that helps to ensure that your upgrade completes successfully.

- About AutoUpgrade Utility System Checks
  To help ensure that your upgrade is successful, Oracle strongly recommends that you check your system using the AutoUpgrade Utility in Analyze mode.
- Example of Running AutoUpgrade Prechecks Using Analyze Mode
  To see how you can use AutoUpgrade to check a non-CDB Oracle Database before an upgrade, use this example to understand the procedure.
- Checking the Upgrade Checks Overview File
  Learn how to use the AutoUpgrade Upgrade Checks Overview file to prepare for your upgrade.
- Creating a Configuration File to Run AutoUpgrade Prechecks On a CDB
  See how you can control how AutoUpgrade performs upgrade prechecks to include or exclude PDBs on a multitenant architecture Oracle Database.
- Running AutoUpgrade Fixups on the Earlier Release Oracle Database
  Use this example to see how to run the AutoUpgrade Fixups that the Analyze mode generates for your system.

About AutoUpgrade Utility System Checks

To help ensure that your upgrade is successful, Oracle strongly recommends that you check your system using the AutoUpgrade Utility in Analyze mode.

To use the AutoUpgrade Utility for your upgrade, you must first run AutoUpgrade in Analyze Mode.

> **Note:**
>
> The AutoUpgrade Utility requires Java 8. To ensure that the correct Java version is available, Oracle recommends that you run AutoUpgrade using the Java version in the target Oracle Database Oracle home. The path to the Java version in the Oracle home is *Oracle-home*/jdk/bin/java, where *Oracle-home* is the target Oracle Database Oracle home. To check the Oracle home has Java 8, you can run the following command:
>
> ```
> $ORACLE_HOME/jdk/bin/java -version
> java version "1.8.0_211"
> Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
> Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed
> mode)
> ```

AutoUpgrade and the Analyze Mode

The AutoUpgrade Utility includes extensive system checks that can help to prevent many issues that can arise during an upgrade. The utility is located in the new Oracle Database binary home. However, to obtain the latest updates, Oracle strongly recommends that you download the most recent version of the tool from My Oracle Support Document 2485457.1. You can place the downloaded file in any directory. To run Analyze to check readiness of the database for upgrade to the new release while your database is running on your earlier release you must specify the target version manually in your configuration file, using the `target_version` parameter. For example: `upg1.target_version=19`.

When you run AutoUpgrade in Analyze mode, AutoUpgrade only reads data from the database, and does not perform any updates to the database. You can run AutoUpgrade using the Analyze mode during normal business hours. You can run AutoUpgrade in Analyze mode on your source Oracle Database home before you have set up your target release Oracle Database home.

Related Topics

- My Oracle Support Note 2485457.1

Example of Running AutoUpgrade Prechecks Using Analyze Mode

To see how you can use AutoUpgrade to check a non-CDB Oracle Database before an upgrade, use this example to understand the procedure.

To use AutoUpgrade, you must have Java 8 installed. Oracle Database Release 12.1 (12.1.0.2) or newer Oracle homes have a valid java version by default. Start AutoUpgrade in Analyze mode using the following syntax, where *Oracle_home* is the Oracle home directory, or the environment variable set for the Oracle home, and `yourconfig.txt` is your configuration file:

```
java -jar autoupgrade.jar -config yourconfig.txt -mode analyze
```

While AutoUpgrade is running, if you want to obtain an overview of the progress, you can enter the command `lsj` on Linux and Unix systems. When all checks are

completed the tool will write the and write the Preupgrade Fixup HTML File, which provides a report on system readiness, display the status of jobs run to the screen, and exit. If all jobs are listed as "finished successfully," then it means that you can go ahead and upgrade the database. However, to see if there are recommendations that you want to follow before starting the upgrade, Oracle still recommends that you look at the Preupgrade Fixup HTML File. If any job is listed as "failed," then it means that there is an error that prevents the upgrade from starting.

Example 2-4 Using AutoUpgrade in Analyze Mode to Check an Oracle Database 12c Non-CDB System

This example shows running AutoUpgrade using a configuration file for upgrading from a Non-CDB Oracle Database 12c Release 2 (12.2) system to a new Oracle Database release, with AutoUpgrade downloaded to the folder `/tmp`:

```
java -jar /tmp/autoupgrade.jar -config 122-to-new.txt -mode analyze
```

The configuration file called for this check (`122-to.new.txt`) is as follows:

```
#12.2-to-19c config file
#
global.autoupg_log_dir=/home/oracle/autoupgrade
upg1.source_home=/u01/app.oracle/product/12.2.0.1
upg1.target_home=/u01/app/oracle/product/19
upg1.sid=dbsales
upg1.start_time=now

upg1.log_dir=/home/oracle/autoupgrade/dbsales
upg1.upgrade_node=localhost
```

Note that the `pdbs` parameter is not specified. You only need to specify the `pdbs` parameter when you want to indicate specific PDBs that you want to check, and exclude other PDBs on the CDB. The value you specify for `log_dir` is the location where AutoUpgrade places the Preupgrade Fixup HTML File. The file is written using the following format, where `log-path` is the path you specify for log files, *`sid`* is the Oracle Database system identifier, and *`job-number`* is the AutoUpgrade job number:

`/log-path/sid/job-number/``prechecks`

When you run AutoUpgrade, the output appears as follows:

```
AutoUpgrade tool launched with default options
+--------------------------------+
| Starting AutoUpgrade execution |
+--------------------------------+
1 databases will be analyzed
Type 'help' to list console commands
upg>
.
.
.
Job 100 completed
```

```
------------------- Final Summary -------------------
Number of databases            [1]

Jobs finished successfuly      [1]
Jobs failed                    [0]
Jobs pending                   [0]
------------- JOBS FINISHED SUCCESSFULLY -------------
Job 100 FOR DB12
```

In this case, the configuration file specifies that you want the log file placed in the path `/home/oracle/autoupgrade`. Because the Oracle Database system identifier (SID) is `CDB1`, and the AutoUpgrade Job is `100`, the Upgrade Checks Overview file for this job is placed in the path `/home/oracle/autoupgrade/dbsales/100/prechecks`.

Review the Upgrade Checks Overview file, and correct any errors that are reported before proceeding with the upgrade. You can run AutoUpgrade in Fixup mode to correct many errors.

With CDBs, you can use the same procedure and the same configuration file.

> **Note:**
>
> When you run AutoUpgrade in Analyze or Fixup mode on a CDB, AutoUpgrade opens all PDBs in the CDB to complete the action. If a PDB is closed, then AutoUpgrade opens the PDB, and leaves it in an OPEN state after the analysis or fixup is completed. If you want to leave PDBs closed, and not perform checks or fixups, then you can specify that only particular PDBs are checked or fixed by using the configuration file parameter `pdb` to list the PDBs that you want AutoUpgrade to check.

Checking the Upgrade Checks Overview File

Learn how to use the AutoUpgrade Upgrade Checks Overview file to prepare for your upgrade.

When the AutoUpgrade Analyze mode is complete, it places files in the directory that you specify with the configuration file `log_dir` parameter. The file is written using the following format, where `log-path` is the path you specify for log files, `sid` is the Oracle Database system identifier (SID), and `job-number` is the AutoUpgrade job number:

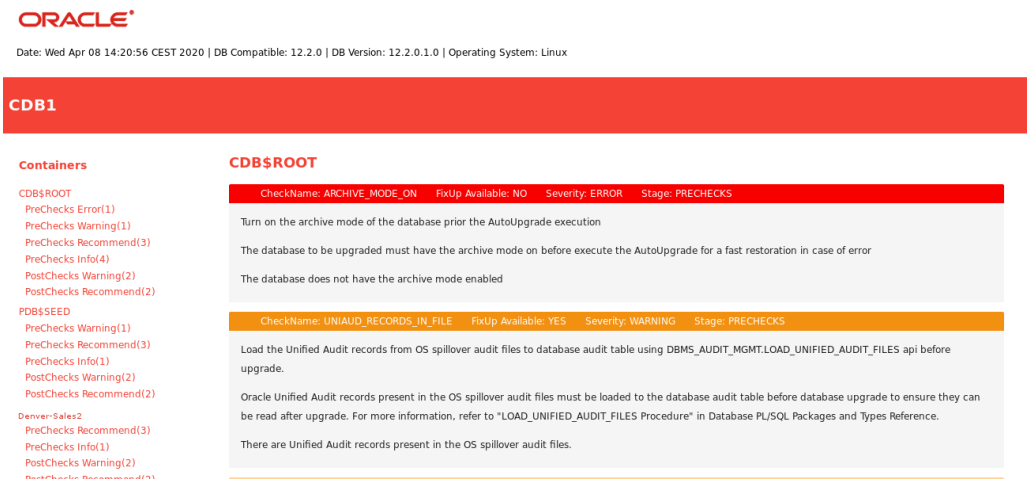
/log-path/sid/job-number/`prechecks`

For example, with the log directory specified as /home/oracle/autoupgrade/DB12, with the database SID `DB12`, and with the job number 100:

```
/home/oracle/autoupgrade/DB12/DB12/100/prechecks
```

Review the Upgrade Checks Overview file, and correct any errors that are reported before proceeding with the upgrade. You can also run AutoUpgrade in Fixup mode to correct many errors.

Figure 2-1 Example of the Upgrade Checks Overview File



In the topic area of the file the results file example shows two report messages for `CDB$ROOT`, which show the name of the check, whether or not a fixup is available that can be run using AutoUpgrade in Fixup mode, the severity of the issue, and the stage of AutoUpgrade that is being run.

Creating a Configuration File to Run AutoUpgrade Prechecks On a CDB

See how you can control how AutoUpgrade performs upgrade prechecks to include or exclude PDBs on a multitenant architecture Oracle Database.

To check Oracle Database servers configured with multitenant container databases (CDBs) and pluggable databases (PDBs), you can use the same procedure and configuration file that you use with a non-CDB Oracle Database. As AutoUpgrade runs checks during an Analyze or Fixup mode run, all of the PDBs in the CDB are opened. If you run AutoUpgrade on a CDB, and a PDB is closed, then AutoUpgrade opens the PDB, and AutoUpgrade leaves it open after Analyze checks or Fixup actions.

If you want to manage which PDBs are opened for checks, so that you can keep some PDBs closed, then you can use the configuration file option `pdbs` to provide a list that includes only the PDBs that you want to be checked. When you provide a list of PDBs to check, AutoUpgrade checks `CDB$ROOT`, `PDB$SEED`, and all of the PDBs that you specify in the list. In this example, the PDB named `denver-sales2` is specified.

Example 2-5 AutoUpgrade Configuration File for a CDB and PDBs

The following example specifies that only the PDB named `denver-sales2` is opened and analyzed.

```
global.autoupg_log_dir=/home/oracle/autoupgrade
upg1.source_home=/u01/app/oracle/product/12.2.0
upg1.target_home=/u01/app/oracle/product/19
upg1.sid=CDB1
upg1.start_time=now
```

```
upg1.log_dir=/home/oracle/autoupgrade/CDB1
upg1.pdbs=denver-sales2
```

Running AutoUpgrade Fixups on the Earlier Release Oracle Database

Use this example to see how to run the AutoUpgrade Fixups that the Analyze mode generates for your system.

When you run AutoUpgrade in Fixup mode, AutoUpgrade performs the checks that it also performs in Analyze mode. After completing these checks, AutoUpgrade then performs all automated fixups that are required for the new release before you start an upgrade. When you plan to move your database to a new release, using the Fixup mode prepares the database for upgrade.

> **Caution:**
>
> Oracle recommends that you run AutoUpgrade in Analyze mode separately before running AutoUpgrade in Fixup mode. Fixup mode can make changes to the source database.

As part of upgrade preparation, if the source database requires corrections for conditions that would cause errors during an upgrade, then AutoUpgrade run in Fixup mode performs automated fixes to the source database. Because running AutoUpgrade in Fixup mode is a step that you perform as you are moving to another system, it does not create a guaranteed restore point. Oracle recommends that you run this mode outside of normal business hours.

```
$ java -jar autoupgrade.jar -config yourconfig.txt -mode fixup
```

If Java 8 is in your source Oracle home, then start AutoUpgrade in Fixup mode using the following syntax, where *`Oracle_home`* is the Oracle home directory, or the environment variable set for the Oracle home, and `yourconfig.txt` is your configuration file:

```
$ java -jar autoupgrade.jar -config yourconfig.txt -mode fixup
```

Testing the Upgrade Process for Oracle Database

Your test plan for Oracle Database upgrades should include these test procedures.

Oracle recommends that you create a full working copy of your database environment in which to test all the pre-upgrade, upgrade, and post-upgrade processes.

You can create a test environment that does not interfere with the current production Oracle database. Oracle Data Guard, for example, enables you to create physical and snapshot standby databases.

Your test environment depends on the upgrade method you choose:

- If you plan to use DBUA or perform a manual upgrade, then create a test version of the current production database.

- If you plan to use Data Pump Export/Import, then export and import in stages, using subsets of the current production database.

Practice upgrading the database using the test environment. The best practice is to perform testing of the upgrade process on an exact copy of the database that you want to upgrade, rather than on a downsized copy or test data. If an exact copy is impractical, then carefully chose a representative subset of your data to move over to your test environment and test the upgrade on that data.

- Example of Testing Upgrades Using Priority List Emulation
  You can use the Parallel Upgrade Utility on multitenant architecture Oracle Databases to run upgrade emulations to test your priority list or other parameter settings before you run your upgrade.
- Upgrade Oracle Call Interface (OCI) and Precompiler Applications
  Upgrade any Oracle Call Interface (OCI) and precompiler applications that you plan to use with the new release of Oracle Database.

See Also:

- *Oracle Database Testing Guide* for information about testing a database upgrade
- *Oracle Database Utilities* for information on Data Pump Export and Import utilities
- *Oracle Data Guard Concepts and Administration* for information on physical and snapshot standby databases

Example of Testing Upgrades Using Priority List Emulation

You can use the Parallel Upgrade Utility on multitenant architecture Oracle Databases to run upgrade emulations to test your priority list or other parameter settings before you run your upgrade.

On multitenant architecture Oracle Database systems, starting with Oracle Database 12*c* release 2 (12.2), you can use priority lists to upgrade or exclude specific PDBs, or to set a specific upgrade priority order. Running the Parallel Upgrade Utility using priority emulation is a way to test your priority list without actually running the upgrade. Use the Parallel Upgrade Utility emulation feature to test your upgrade plan using priority lists.

Preparing for Upgrade Emulation Tests

Before you run the emulation, you must set up your source and target upgrade locations, and prepare your database in the same way you prepare for an actual upgrade. No upgrade actually occurs, but the Parallel Upgrade Utility generates log files that show how an actual upgrade is carried out.

> **Note:**
>
> You can use the `-E` parameter to run the Parallel Upgrade Utility in emulation mode to test how priority lists run, or to test how other upgrade parameter selections are carried out during an upgrade. For example, you can run an upgrade emulation to obtain more information about how the resource allocation choices you make using the `-n` and `-N` parameters are carried out.

Syntax for Running Priority List Emulation

You can use any of the parameter settings that you normally use with the Parallel Upgrade Utility, However, you must create a priority list, and you must use the `-L` parameter to call the list when you run the Parallel Upgrade Utility with the `-E` parameter to set it to perform an upgrade emulation.

The following is an example of the minimum required syntax for running the Parallel Upgrade Utility using priority list emulation, where *`priority_list_name`* is the name of your priority list file:

```
catctl -E -L priority_list_name catupgrd.sql
```

Example 2-6 Example of Running the Parallel Upgrade Utility using Priority List Emulation

The following example uses this priority list, which is named `plist.txt`:

```
1,CDB$ROOT
2,PDB$SEED
3,CDB1_PDB2
4,CDB1_PDB4
4,CDB1_PDB3
5,CDB1_PDB5
5,CDB1_PDB1
```

The following command runs a parallel emulation, calling this priority list:

```
$ORACLE_HOME/perl/bin/perl catctl.pl -L plist.txt -E -n 4 -N 2
catupgrd.sql
```

This command uses the following parameter settings:

- `-E` specifies that Parallel Upgrade Utility runs the upgrade procedures in emulation mode.
- `-n 4` specifies that the upgrade allocates four processes to perform parallel upgrade operations.
- `-N 2` specifies that the upgrade runs two SQL processors to upgrade the PDBs. The maximum PDB upgrades running concurrently is the value of `-n` divided by the value of `-N`, so the upgrade runs no more than two concurrent PDB upgrades.
- `-L` specifies the priority list that the command reads to set upgrade priority.

As the Parallel Upgrade Utility carries out the emulated upgrade, it displays on screen the same output that you see during an actual upgrade.

When the upgrade emulation completes, it generates a log file, `catctl_prority_run.list`, which is stored either in the default logging directory, or in a logging directory location that you specify with the `-l` parameter. Because in this example we did not specify a different log directory, and we are running the upgrade on the container database named `CDB1`, the output is place in the path *`Oracle_base`*`/cfgtoollogs/CDB1/run`, where *`Oracle_base`* is the Oracle base of the user running the upgrade, and `CDB1` is the name of the container database (CDB) on which you are running the upgrade.

The log file `catctl_priority_run.lst` displays the list of how the upgrade priority was carried out during the upgrade emulation. It shows how the Parallel Upgrade Utility grouped PDB upgrades. You can see a priority run that contains the groupings and priorities before you actually carry out the upgrade. The log file generated by the upgrade is also displayed on the screen after the upgrade completes.

At the conclusion of the upgrade log, the log will show that `CDB$ROOT` is upgraded first. After the CDB$ROOT upgrade is completed, the Parallel Upgrade Utility carries out the following concurrent upgrades of PDBs, in accordance with the priority settings in the priority list:

1. `PDB$SEED` and `CDB1_PDB2`. Output logs are generated with log Identifiers (Log IDs) specified as `pdb_seed` for `PDB$SEED`, and log ID `mayapdb2` for `CDB_1PDB2`)
2. `CDB1_PDB3` and `CDB1_PDB4`. Log IDs are specified `mayapdb3` and `mayapdb4`
3. `CDB1_PDB5` and `CDB1_PDB6`. Log IDs are specified `mayapdb5` and `mayapdb6`
4. `CDB1_PDB1`. The log ID is specified as `mayapdb1`.

Related Topics

- About the Parallel Upgrade Utility for Oracle Database (CATCTL.PL and DBUPGRADE)
- Parallel Upgrade Utility (catctl.pl) Parameters

Upgrade Oracle Call Interface (OCI) and Precompiler Applications

Upgrade any Oracle Call Interface (OCI) and precompiler applications that you plan to use with the new release of Oracle Database.

Oracle recommends that you test these applications on a test database before you upgrade your current production database.

Related Topics

- About Upgrading Precompiler and OCI Applications in Oracle Database

Requirements for Upgrading Databases That Use Oracle Label Security and Oracle Database Vault

You must complete these tasks before starting an upgrade with a database using Oracle Label Security or Oracle Database Vault.

- Audit Table Preupgrade and Archive Requirements
  For Oracle Database releases earlier than 12.1 using Oracle Label Security and Oracle Database Vault, you must run the OLS preprocess script before you upgrade.
- DBUA, AutoUpgrade, and Oracle Database Vault
  If you use Database Upgrade Assistant (DBUA) to upgrade Oracle Database 12c Release 2 (12.2), Oracle Database 18c, or Oracle Database 19c, and your system uses Oracle Database Vault, then you must use this upgrade process.
- Oracle Database Vault and Upgrades of Oracle Database Release 11.2
  If Oracle Label Security is installed in the earlier release that you are upgrading, then grant the DV_PATCH_ADMIN role to SYS.
- Granting the DV_PATCH_ADMIN Role to SYS for Oracle Database Vault
  If Oracle Database Vault is enabled, then to perform checks for Oracle Data Vault, the upgrade process requires running three SQL scripts.

Audit Table Preupgrade and Archive Requirements

For Oracle Database releases earlier than 12.1 using Oracle Label Security and Oracle Database Vault, you must run the OLS preprocess script before you upgrade.

If you are upgrading from a database earlier than Oracle Database release 12.1 that uses Oracle Label Security (OLS) and Oracle Database Vault, then you must first run the OLS preprocess script, `olspreupgrade.sql`, to process the `aud$` table contents. The OLS upgrade moves the `aud$` table from the `SYSTEM` schema to the `SYS` schema. The `olspreupgrade.sql` script is a preprocessing script required for this move.

Caution:

Running the `olspreupgrade.sql` script before upgrading is mandatory for upgrading databases earlier than Oracle Database release 12.1 that use Oracle Label Security and Oracle Database Vault. Once you have upgraded to Oracle Database release 12.1, you do not have to perform the OLS preprocessing procedure going forward to patch or upgrade the database.

The `olspreupgrade.sql` script creates a temporary table `PREUPG_AUD$` in the `SYS` schema and moves the `SYSTEM.aud$` records to `SYS.PREUPG_AUD$`. As a safety measure, Oracle recommends that you archive your audit trail before running the `olspreupgrade.sql` script. If Oracle Label Security is installed on your database, and you are upgrading from an earlier release, then you must run the OLS preprocess script before upgrading.

See Also:

Oracle Database Security Guide for information about archiving audit trails

Oracle Label Security Administrator's Guide for information about the OLS preprocess script

DBUA, AutoUpgrade, and Oracle Database Vault

If you use Database Upgrade Assistant (DBUA) to upgrade Oracle Database 12c Release 2 (12.2), Oracle Database 18c, or Oracle Database 19c, and your system uses Oracle Database Vault, then you must use this upgrade process.

If you use DBUA to perform the upgrade, then DBUA uses the AutoUpgrade utility to complete the upgrade. However, in that case, AutoUpgrade does not automatically disable Oracle Data Vault for upgrades from Oracle Database 12c Release 2 (12.2), Oracle Database 18c (18.3), or Oracle Database 19c (19.3) to the new Oracle Database release. Before you upgrade a source Oracle Database from one of these releases, you must first disable Oracle Data Vault.

Example 2-7 Overview of Upgrade Procedure for Databases Using Oracle Data Vault

You complete an upgrade from a source database where Oracle Database Vault is enabled with the following steps:

1. Disable Oracle Data Vault.
2. Install the new Oracle Database release.
3. Download the latest AutoUpgrade JAR file from My Oracle Support note 2485457.1, and replace the AutoUpgrade JAR file in the new Oracle Database release, in the path *Oracle_home*/rdbms/admin
4. Run the AutoUpgrade utility (or Database Upgrade Assistant), and complete the upgrade.
5. Enable Oracle Database Vault in the upgraded Oracle Database.

Related Topics

- Disabling and Enabling Oracle Database Vault
- My Oracle Support Document 2485457.1

Oracle Database Vault and Upgrades of Oracle Database Release 11.2

If Oracle Label Security is installed in the earlier release that you are upgrading, then grant the DV_PATCH_ADMIN role to SYS.

If Oracle Database Vault is not installed with your release 11.2 database, then you can skip steps 2, 3, 6, and 7 in this section.

To run the OLS preprocess script on a release 11.2 database before upgrading:

1. Copy the following scripts script from the newly installed Oracle home to the Oracle home of the database that y0u want to upgrade:
   - *ORACLE_HOME*/rdbms/admin/olspreupgrade.sql
   - *ORACLE_HOME*/rdbms/admin/emremove.sql
   - *ORACLE_HOME*/rdbms/admin/catnoamd.sql
2. Start SQL*Plus and connect as DVOWNER to the database that you want to upgrade.

3. Run the following statement:

```
SQL> GRANT DV_PATCH_ADMIN to SYS;
```

4. At the system prompt, connect SYS as SYSDBA:

```
CONNECT SYS AS SYSDBA
```

5. Run the preprocess scripts for Data Vault

```
ORACLE_HOME/rdbms/admin/olspreupgrade.sql
ORACLE_HOME/rdbms/admin/emremove.sql
ORACLE_HOME/rdbms/admin/catnoamd.sql
```

   You may continue to run your applications on the database while the preprocess scripts are running.

6. After the `olspreupgrade.sql` completes its run successfully, start SQL*Plus and connect to the database as `DVOWNER`.
7. Run the following SQL statement:

```
SQL> REVOKE DV_PATCH_ADMIN from SYS;
```

Granting the DV_PATCH_ADMIN Role to SYS for Oracle Database Vault

If Oracle Database Vault is enabled, then to perform checks for Oracle Data Vault, the upgrade process requires running three SQL scripts.

The SYS user requires the DV_PATCH_ADMIN role for the following scripts:

OLS_SYS_MOVE runs `olspreupgrade.sql`:

```
CHECK.OLS_SYS_MOVE.MIN_VERSION_INCLUSIVE=10.2
CHECK.OLS_SYS_MOVE.MAX_VERSION_EXCLUSIVE=12.1
```

EM_PRESENT runs `emremove.sql`:

```
CHECK.EM_PRESENT.MIN_VERSION_INCLUSIVE=NONE
CHECK.EM_PRESENT.MAX_VERSION_EXCLUSIVE=12.1.0.1
```

AMD_EXISTS CHECK runs `catnoamd.sql`:

```
CHECK.AMD_EXISTS.MIN_VERSION_INCLUSIVE=NONE
CHECK.AMD_EXISTS.MAX_VERSION_EXCLUSIVE=NONE
```

Backing Up Oracle Database for Upgrading

Use this procedure to back up your existing Oracle Database before you attempt an upgrade.

> **Caution:**
>
> Before you make any changes to the Oracle software, Oracle recommends that you create a backup of the Oracle software and databases. For Oracle software running on Windows operating systems, you must also take a backup of the Windows registry. Without a registry backup, you cannot restore the Oracle software to a working state if the upgrade,fails, and you want to revert to the previous software installation.

If you choose to use the manual Pre-Upgrade Information Tool, then run it before you cleanly shut down the database. To minimize downtime, you can perform an online backup or create a guaranteed restore point. Database Upgrade Assistant (DBUA) enables you to specify your backup and restore point.

1. Sign on to Oracle RMAN:

```
rman "target / nocatalog"
```

2. Run the following RMAN commands:

```
RUN
{
    ALLOCATE CHANNEL chan_name TYPE DISK;
    BACKUP DATABASE FORMAT 'some_backup_directory%U' TAG
before_upgrade;
    BACKUP CURRENT CONTROLFILE FORMAT 'controlfile location and
name';
}
```

> **See Also:**
>
> - *Oracle Database Backup and Recovery User's Guide* for information about online backups and backup mode
> - *Oracle Database Backup and Recovery User's Guide* for information about Flashback Database and restore points
> - *Oracle Database Backup and Recovery User's Guide* for more information about performing RMAN backups

3

Using AutoUpgrade for Oracle Database Upgrades

Learn how to use AutoUpgrade to simplify your upgrade tasks.

- About Oracle Database AutoUpgrade
  The AutoUpgrade utility identifies issues before upgrades, performs pre- and postupgrade actions, deploys upgrades, performs postupgrade actions, and starts the upgraded Oracle Database.
- Checking for Java Version Support
  Typically, you start the AutoUpgrade JAR file in the Oracle home from which you are upgrading. However, the Oracle home must contain Java 8.
- Examples of How to Use AutoUpgrade
  To guide your upgrade, use the AutoUpgrade workflow example that matches your upgrade use case.
- AutoUpgrade Messages and Process Description Terms
  To understand how your upgrade checks and operations are proceeding, learn about the AutoUpgrade utility messages that are generated as the utility runs.
- About AutoUpgrade Processing Modes
  The four AutoUpgrade processing modes (Analyze, Fixup, Deploy, and Upgrade) characterize the actions that AutoUpgrade performs as it runs.
- Understanding AutoUpgrade Workflows and Stages
  The AutoUpgrade workflow automates each step of a typical upgrade process. The stages that run depend on the processing mode that you select.
- Understanding Non-CDB to PDB Upgrades with AutoUpgrade
  You can upgrade and convert a non-CDB to a PDB in a new CDB in a single operation, or upgrade and then convert a Non-CDB database to a PDB in a pre-existing CDB.
- AutoUpgrade Command-Line Parameters
  Review the AutoUpgrade parameters and select the parameters and options for your Oracle Database upgrade use case.
- AutoUpgrade Utility Configuration Files
  AutoUpgrade configuration files contain all the information required to perform Oracle Database upgrades.
- AutoUpgrade and Oracle Database Configuration Options
  When you run AutoUpgrade, it determines the type of database (Oracle Database, Oracle Database Standalone with Oracle ASM, or Oracle RAC), and performs an upgrade for that type of database
- AutoUpgrade Configuration File Examples
  Use these examples to understand how you can modify your own AutoUpgrade configuration files to perform a variety of configuration actions during the upgrade.
- AutoUpgrade Internal Settings Configuration File
  Internal configuration settings control how AutoUpgrade runs.

- AutoUpgrade Log File Structure
  The AutoUpgrade utility produces a log file structure that includes job status and configuration files.
- Enabling Full Deployments for AutoUpgrade
  To enable a guaranteed restore point (GRP) so that you can flashback an upgrade, you must set up archive logging, and you should complete other tasks to enable AutoUpgrade to complete the upgrade.
- Examples of How to Use the AutoUpgrade Console
  The AutoUpgrade console provides a set of commands to monitor the progress of AutoUpgrade jobs. The console starts by default when you run the AutoUpgrade utility, and is enabled or disabled by the parameters `console` and `noconsole`.
- How to Override Default Fixups
  You can use the `RUNFIX` column entry to disable automated fixups, except in cases where disabling the fixup violates security or Oracle policy.
- Local Configuration File Parameter Fixups Checklist Example
  To include or exclude specific fixups for individual databases during upgrades, use the local configuration file checklist.
- Proper Management of AutoUpgrade Database Changes
  AutoUpgrade is a powerful utility, which requires that you use it responsibly. Review and avoid using AutoUpgrade in ways that put the database at risk.
- AutoUpgrade and Microsoft Windows ACLs and CLIs
  When running AutoUpgrade on Microsoft Windows systems, Oracle recommends additional best practices with access control lists (ACLs) and command-line interfaces (CLIs).

About Oracle Database AutoUpgrade

The AutoUpgrade utility identifies issues before upgrades, performs pre- and postupgrade actions, deploys upgrades, performs postupgrade actions, and starts the upgraded Oracle Database.

The AutoUpgrade utility is designed to automate the upgrade process, both before starting upgrades, during upgrade deployments, and during postupgrade checks and configuration migration. You use AutoUpgrade after you have downloaded binaries for the new Oracle Database release, and set up new release Oracle homes. When you use AutoUpgrade, you can upgrade multiple Oracle Database deployments at the same time, using a single configuration file, customized as needed for each database deployment.

With the January 2019 Release Updates (`DBJAN2019RU`) and later updates, AutoUpgrade support is available for Oracle Database 12c Release 2 (12.2) and Oracle Database 18c (18.5) target homes. For both Oracle Database 12c Release 2 (12.2) and Oracle Database 18c (18.5) target homes, you must download the AutoUpgrade kit from My Oracle Support Document 2485457.1.

In Oracle Database 19c (19.3) and later target Oracle homes, the `autoupgrade.jar` file exists by default. However, before you use AutoUpgrade, Oracle strongly recommends that you download the latest AutoUpgrade version. AutoUpgrade is included with each release update (RU), but the most recent AutoUpgrade version is always available from My Oracle Support Document 2485457.1.

> **Note:**
>
> AutoUpgrade is available for Oracle Database Enterprise Edition, and Oracle Database Standard Edition. It is not available for Oracle Database Express Edition.

Preventing Issues: Analyze and Fixup Modes

Before the upgrade, in Analyze mode, the AutoUpgrade utility performs read-only analysis of databases before upgrade, so that it can identify issues that require fixing. You can run the utility during normal database operations. In Fixup Mode, the AutoUpgrade utility detects and identifies both fixes that require manual intervention, and fixes that the AutoUpgrade utility can perform during the upgrade deployment phase.

Simplifying Upgrades: Deploy and Upgrade Modes

In Deploy phase, the AutoUpgrade utility modifies the databases you indicate in your configuration file. It enables you to call your own custom scripts during the upgrade to configure databases. In many cases, the AutoUpgrade utility can perform automatic fixes to databases during the upgrade process without requiring manual intervention.

Deploy and Upgrade Postupgrade Checks and Fixes

After an upgrade completes with either Deploy or Upgrade modes, AutoUpgrade performs postupgrade checks. It provides a process where you can enable your custom scripts to be run on each of the upgraded databases, in accordance with the configuration instructions you provide in the AutoUpgrade configuration file, and also can run automatic postupgrade fixups as part of the postupgrade process. In Deploy mode, AutoUpgrade also confirms that the upgrade has succeeded, and copies database files such as `sqlnet.ora`, `tnsname.ora`, and `listener.ora` from the source home to the target home. After these actions are complete, the upgraded Oracle Database release is started in the new Oracle home.

Related Topics

- My Oracle Support Document 2485457.1

Checking for Java Version Support

Typically, you start the AutoUpgrade JAR file in the Oracle home from which you are upgrading. However, the Oracle home must contain Java 8.

The AutoUpgrade utility is a Java 8 release JAR file, which is available in Oracle Database Oracle homes in releases after Oracle Database 12c Release 1 (12.1.0.2). On Oracle Database releases 12.2, 18 and 19, Java 8 is the default. The path to the Java version in the Oracle home is *`Oracle-home`*`/jdk/bin/java`, where *`Oracle-home`* is the Source database Oracle home (the earlier Oracle Database release that you are upgrading). For example:

```
$ORACLE_HOME/jdk/bin/java -version
```

The result should be a Java 8 version. For example:

```
$ORACLE_HOME/jdk/bin/java -version
java version "1.8.0_251"
Java(TM) SE Runtime Environment (build 1.8.0_251-b08)
```

A successful response to `autoupgrade.jar -version` should be similar to the following:

```
java -jar ./build/production/autoupgrade.jar -version
build.hash 2ce58db
build.version 19.7.4
build.date 2020/01/17 10:54:33
build.max_target_version 19
build.type production
```

If your you attempt to start AutoUpgrade with a Java release earlier than Java 8. then the following error occurs:

```
$ORACLE_HOME/jdk/bin/java -jar autoupgrade.jar -version
Exception in thread "main" java.lang.UnsupportedClassVersionError: Bad
version number in .class file
    at java.lang.ClassLoader.defineClass1(Native Method)
    at ...
```

Examples of How to Use AutoUpgrade

To guide your upgrade, use the AutoUpgrade workflow example that matches your upgrade use case.

These examples are presented in a typical workflow sequence. To see how you can use the configuration file to run scripts with the `noconsole` parameter, see examples under "How to Use the AutoUpgrade Console."

- AutoUpgrade with Source and Target Database Homes on Same Server (Typical)
  When your Oracle Database Source and Target Oracle homes are installed on the same physical server, use this example.
- AutoUpgrade with Source and Target Database Homes on Different Servers
  When your Oracle Database Source and Target Oracle homes are located on different physical servers, you must complete tasks on both servers.

Related Topics

- Examples of How to Use the AutoUpgrade Console
  The AutoUpgrade console provides a set of commands to monitor the progress of AutoUpgrade jobs. The console starts by default when you run the AutoUpgrade utility, and is enabled or disabled by the parameters `console` and `noconsole`.

AutoUpgrade with Source and Target Database Homes on Same Server (Typical)

When your Oracle Database Source and Target Oracle homes are installed on the same physical server, use this example.

Context: Source and Target homes are on the same server.

To start the analysis, enter the following command.

```
java -jar autoupgrade.jar -config config.txt -mode analyze
```

The command produces a report that indicates any error conditions that the command finds. Review the error conditions. When relevant, customize the fixups that AutoUpgrade has created to address error conditions.

To start the deployment of the upgrade, enter the following command:

```
java -jar autoupgrade.jar -config config.txt -mode deploy
```

AutoUpgrade with Source and Target Database Homes on Different Servers

When your Oracle Database Source and Target Oracle homes are located on different physical servers, you must complete tasks on both servers.

Context: Source and Target Oracle homes are on different physical servers.

To start the analysis, enter the following command.

```
java -jar autoupgrade.jar -config config.txt -mode analyze
```

The command produces a report that indicates any error conditions that the command finds. Review the error conditions. When relevant, customize the fixups that AutoUpgrade has created to address error conditions.

Because the source and target Oracle Database Oracle homes are on different servers, you run fixups on each server.

1. Run fixups on the source server:

   ```
   java -jar autoupgrade.jar -config config.txt -mode fixups
   ```

2. Complete the tasks to move the source Oracle Database from the source server to the target server.
3. On the target server, start up the database in upgrade mode, and then run AutoUpgrade in `upgrade` mode:

   ```
   java -jar autoupgrade.jar -config config.txt -mode upgrade
   ```

AutoUpgrade Messages and Process Description Terms

To understand how your upgrade checks and operations are proceeding, learn about the AutoUpgrade utility messages that are generated as the utility runs.

- Overview of AutoUpgrade Job IDs
  An AutoUpgrade **Job** is a unit of work associated with an upgrade, which is identified by a job identifier (`jobid`).
- Overview of AutoUpgrade Stages
  AutoUpgrade utility jobs pass through a series of phases, called **stages**, during which specific actions are performed.
- Overview of AutoUpgrade Stage Operations and States
  In AutoUpgrade, **operation** describes actions performed during stages, and **state** indicates the status of a stage operation.

Overview of AutoUpgrade Job IDs

An AutoUpgrade **Job** is a unit of work associated with an upgrade, which is identified by a job identifier (`jobid`).

A job represents a set of actions that AutoUpgrade performs. Each job goes through a set of stages to accomplish its purpose. The job is identified by a unique positive integer, which is called a `jobid`. Each new AutoUpgrade job produces a new job ID (`jobid`) for each database found in the configuration file for the AutoUpgrade utility. If AutoUpgrade detects that a database with a `jobid` that you previously started exists on your system is incomplete, then AutoUpgrade identifies this existing `jobid` as a **resume** operation. In a resume operation, stages of a job identified by a `jobid` that did not complete during the previous AutoUpgrade run are continued from the point where they were stopped.

Overview of AutoUpgrade Stages

AutoUpgrade utility jobs pass through a series of phases, called **stages**, during which specific actions are performed.

The actions that occur during a stage are defined by the **processing mode** that you select for AutoUpgrade: **Analyze**, **Fixups**, **Deploy**, and **Upgrade**.

AutoUpgrade has the following stages:

- **SETUP**: The initial stage that the AutoUpgrade utility job manager creates as part of the preparation for starting a job.
- **PREUPGRADE**: The stage in which AutoUpgrade performs checks of your system, based on your current system configuration to determine its readiness for upgrade, such as checking to determine if you have sufficient available disk space.
- **PRECHECKS**: The stage in which AutoUpgrade analyzes your source Oracle home to determine if the database meets the requirements for upgrade.
- **GRP**: The guaranteed restore point (GRP), which AutoUpgrade creates before starting the upgrade process. This option is only available for Oracle Database Enterprise Edition releases. It is not available for Oracle Database Standard

Edition. Even though AutoUpgrade creates a GRP by default, Oracle highly recommends that you perform a backup before starting your upgrade.

- **PREFIXUPS**: The stage in which AutoUpgrade performs preupgrade fixups before starting the upgrade. For example, this is the stage in which AutoUpgrade gathers dictionary statistics on the source Oracle home.
- **DRAIN**: The stage during which AutoUpgrade shuts down the database.
- **DBUPGRADE**: The stage in which AutoUpgrade performs the upgrade, and compiles any invalid objects that are found after the upgrade completes.
- **POSTCHECKS**: The stage in which AutoUpgrade performs checks on the target Oracle home (the upgraded Oracle Database) before starting postupgrade fixups.
- **POSTFIXUPS**: The stage in which AutoUpgrade performs processing of postupgrade fixups, such as upgrading the time zone.
- **POSTUPGRADE**: The stage in which AutoUpgrade copies or merges the source Oracle home configuration files (`tnsnames.ora`, `sqlnet.ora`, and other files) to the target Oracle home.

Overview of AutoUpgrade Stage Operations and States

In AutoUpgrade, **operation** describes actions performed during stages, and **state** indicates the status of a stage operation.

Understanding Operation Messages

An operation message is an internal phase message that describes what is happening during an AutoUpgrade state. There are two types of operation messages.

PREPARING: An AutoUpgrade instance is being created, initialized, or called, in preparation for completing an AutoUpgrade stage. This is an information message. When you see this message, there is no action for you to perform.

EXECUTING: AutoUpgrade is in the process of performing the main workflow of a stage. This is an information message. There is no action for you to perform.

Understanding State Messages

State messages indicate the status of the current workflow of the stage for which the message is displayed. There are four state messages:

- **ABORTED**: AutoUpgrade stopped performing the stage workflow, in response to a user request.
- **ERROR**: An error was encountered while the stage workflow was being performed. Review the cause of the error.
- **FINISHED**: AutoUpgrade successfully completed the workflow for the stage.
- **RUNNING**: AutoUpgrade is performing the stage workflow.

About AutoUpgrade Processing Modes

The four AutoUpgrade processing modes (Analyze, Fixup, Deploy, and Upgrade) characterize the actions that AutoUpgrade performs as it runs.

- Preparations for Running AutoUpgrade Processing Modes
  You must complete preparations before you can run an AutoUpgrade processing mode.
- About the AutoUpgrade Analyze Processing Mode
  The AutoUpgrade Analyze (`analyze`) processing mode checks your database to see if it is ready for upgrade.
- About the AutoUpgrade Fixups Processing Mode
  The AutoUpgrade Fixups (`fixups`) processing mode analyzes your database, and performs fixups of items that must be corrected before you can perform an upgrade.
- About the AutoUpgrade Deploy Processing Mode
  The AutoUpgrade Deploy (`deploy`) processing mode performs the actual upgrade of the database, and performs any pending fixups.
- About the AutoUpgrade Upgrade Processing Mode
  The AutoUpgrade Upgrade (`upgrade`) processing mode enables you to upgrade the target Oracle home in cases where you do not have access to the source Oracle home.

Preparations for Running AutoUpgrade Processing Modes

You must complete preparations before you can run an AutoUpgrade processing mode.

Before you can use an AutoUpgrade processing mode, confirm that you meet the following requirements:

- You have created a user configuration file.
- The source Oracle Database release is up and running in the original Oracle home. In case of a restart of AutoUpgrade, you must start the database in the Oracle home that corresponds to the phase in the upgrade flow.
- The server on which the database is running is registered on the server hosts file (for example, `/etc/hosts`), or on a domain name server (DNS).

  If you are logged in to the server on which the target database is located, and the database is running either on `localhost`, or where AutoUpgrade is running, then remove the hostname parameter from the AutoUpgrade `config` file.
- On container databases (CDBs), if you want to upgrade a subset of pluggable databases (PDBs), then the PDBs on which you want to run the upgrade are open, and they are configured in the user configuration file, using the AutoUpgrade local parameter `pdbs`. If you do not specify a list of PDBs, then AutoUpgrade upgrades all PDBs on the CDB.
- You have the AutoUpgrade jar file (`autoupgrade.jar`) downloaded or available, and you are able to run it using a Java 8 distribution.
- If you want to run AutoUpgrade in a batch or script , then you have called AutoUpgrade using the `noconsole` parameter in the command.

In Oracle Database 19c (19.3) and later target Oracle homes, the `autoupgrade.jar` file exists by default. However, before you use AutoUpgrade, Oracle strongly recommends that you download the latest version, which is available form My Oracle Support Document 2485457.1.

Related Topics

- My Oracle Support Document 2485457.1

About the AutoUpgrade Analyze Processing Mode

The AutoUpgrade Analyze (`analyze`) processing mode checks your database to see if it is ready for upgrade.

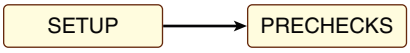
When you run AutoUpgrade in Analyze mode, AutoUpgrade only reads data from the database, and does not perform any updates to the database. You can run AutoUpgrade using the Analyze mode during normal business hours. You can run AutoUpgrade in Analyze mode on your source Oracle Database home before you have set up your target release Oracle Database home.

You start AutoUpgrade in Analyze mode using the following syntax, where *`Java-8-home`* is the location of your Java 8 distribution, or the environment variable set for the Java 8 home:

```
Java-8-home/bin/java -jar autoupgrade.jar -config yourconfig.txt -mode
analyze
```

Oracle Database Release 12.2 (12.2.0.1) or newer Oracle homes have a valid java version by default. Start AutoUpgrade in Analyze mode using the following syntax, where *`Oracle_home`* is the Oracle home directory, or the environment variable set for the Oracle home, and `yourconfig.txt` is your configuration file:

```
Oracle_home/jdk8/bin/java -jar autoupgrade.jar -config yourconfig.txt
-mode analyze
```



The AutoUpgrade Analyze mode produces two output files, which are given the name of the system identifier (*`SID`*) of the database that you check:

- *`SID`*`.html`: View this file using a web browser.
- *`SID`*`_preupgrade.log`: View this file using a text editor.

Each report identifies upgrade errors that would occur if you do not correct them, either by running an automatic fixup script, or by manual correction. If errors occur, then they are reported in the user log file, and also in the `status.json` file.

The Analyze mode also generates a status directory in the path `cfgtoollogs/upgrade/auto/status`. This directory contains files that indicate if the analysis was successful or failed. This directory has two JSON files, `status.json` and `progress.json`:

- `status.json` : A high-level status JSON file that contains the final status of the upgrade.
- `progress.json`: A JSON file that contains the current progress of all upgrades being performed on behalf of the configuration file. If errors occur, then they are reported in the log file of the user running AutoUpgrade, and also in the `status.json` file.

If your target database Oracle home is not available on the server, then in your configuration file, you must set the source Oracle home parameters to the same path, so that the AutoUpgrade analyze processing mode can run. For example:

```
#
# Source Home
#
sales3.source_home=d:\app\oracle\product\12.1.0\dbhome_1
#
# Target Oracle Home
#
sales3.target_home=d:\app\oracle\product\12.1.0\dbhome_1
```

Earlier releases of AutoUpgrade required you to set `target_home`. In later releases of AutoUpgrade, this restriction has been lifted for both Analyze and Fixups modes.

About the AutoUpgrade Fixups Processing Mode

The AutoUpgrade Fixups (`fixups`) processing mode analyzes your database, and performs fixups of items that must be corrected before you can perform an upgrade.

When you run AutoUpgrade in Fixups mode, AutoUpgrade performs the checks that it also performs in Analyze mode. After completing these checks, AutoUpgrade then performs all automated fixups that are required to fix before you start an upgrade. When you plan to move your database to a different platform, using the Fixups mode prepares the database for upgrade.

> **Caution:**
>
> Oracle recommends that you run AutoUpgrade in Analyze mode separately before running AutoUpgrade in Fixups mode. Fixup mode can make changes to the source database.

As part of upgrade preparation, if the source database requires corrections for conditions that would cause errors during an upgrade, then AutoUpgrade run in Fixups mode performs automated fixes to the source database. Because running AutoUpgrade in Fixups mode is a step that you perform as you are moving to another system, it does not create a guaranteed restore point. Oracle recommends that you run this mode outside of normal business hours.

You start AutoUpgrade in Fixups mode using the following syntax, where *`Java-8-home`* is the location of your Java 8 distribution, or the environment variable set for the Java 8 home:

```
Java-8-home/bin/java -jar autoupgrade.jar -config yourconfig.txt -mode
fixups
```

If Java 8 is in your source Oracle home, then start AutoUpgrade in Fixups mode using the following syntax, where *`Oracle_home`* is the Oracle home directory, or

the environment variable set for the Oracle home, and `yourconfig.txt` is your configuration file:

```
Oracle_home/jdk8/bin/java -jar autoupgrade.jar -config yourconfig.txt
-mode fixups
```

As AutoUpgrade runs in Fixups mode, it starts out by running the same prechecks that are run in Analyze mode. It then runs automated fixups in the source database in preparation for upgrade, and generates a high-level status file that indicates the success or failure of fixup operations. If errors occur, then they are reported in the log file of the user running AutoUpgrade.

> **Caution:**
>
> AutoUpgrade in Fixups mode does not create a guaranteed restore point. Before starting AutoUpgrade in Fixups mode, ensure that your database is backed up.

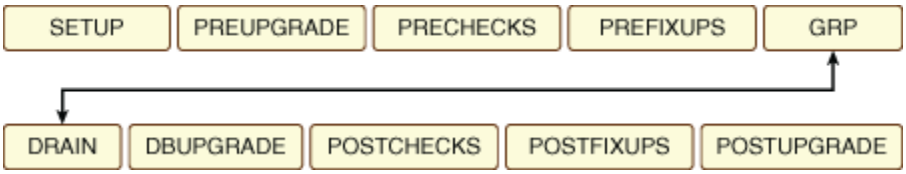
About the AutoUpgrade Deploy Processing Mode

The AutoUpgrade Deploy (`deploy`) processing mode performs the actual upgrade of the database, and performs any pending fixups.

Before you run Deploy, you must have the target Oracle home already installed, and you must have a backup plan in place, in addition to the backup plan run as part of the AutoUpgrade script.

You start AutoUpgrade in Deploy mode using the following syntax, where *`Oracle_home`* is the Oracle home directory, or the environment variable set for the Oracle home, and `yourconfig.txt` is your configuration file:

```
Oracle_home/jdk8/bin/java -jar autoupgrade.jar -config yourconfig.txt
-mode deploy
```



When you run AutoUpgrade in Deploy mode, AutoUpgrade runs all upgrade operations on the database, from preupgrade source database analysis to post-upgrade checks. Each operation prepares for the next operation. If errors occur, then the operation is stopped. All errors are logged to relevant log files, and to the console, if enabled. A high level status file is generated for each operation, which shows the success or failure of the operation. If there are fixups that are still pending (for example, if you run AutoUpgrade in Deploy mode without running AutoUpgrade first in Analyze and Fixups mode) then AutoUpgrade can complete fixups during the Deploy mode.

About the AutoUpgrade Upgrade Processing Mode

The AutoUpgrade Upgrade (`upgrade`) processing mode enables you to upgrade the target Oracle home in cases where you do not have access to the source Oracle home.

You can use the Upgrade mode to divide an upgrade into two parts: First, if you choose, you can run the prefixups mode on the source database running on its Oracle home. Second, you use the Upgrade mode to perform the upgrade of the database on its new home. To use the Upgrade mode, the database must be up and running in the target Oracle home before you run AutoUpgrade in Upgrade mode.

You start AutoUpgrade in Upgrade mode using the following syntax, where *`Oracle_home`* is the Oracle home directory, or the environment variable set for the Oracle home, and `yourconfig.txt` is your configuration file:

```
Oracle_home/jdk8/bin/java -jar autoupgrade.jar -config yourconfig.txt
-mode upgrade
```

This command runs the upgrade and postupgrade operations on the database. Use this option when you have moved your Oracle Database to a different system from the original source system, so that you cannot use the AutoUpgrade Deploy mode.

As AutoUpgrade runs in Upgrade mode, errors are logged to the log file of the user running the AutoUpgrade script. A high level status file is generated for each operation, which shows the success or failure of the operation. When you run AutoUpgrade in Upgrade mode, no postupgrade operations are performed.

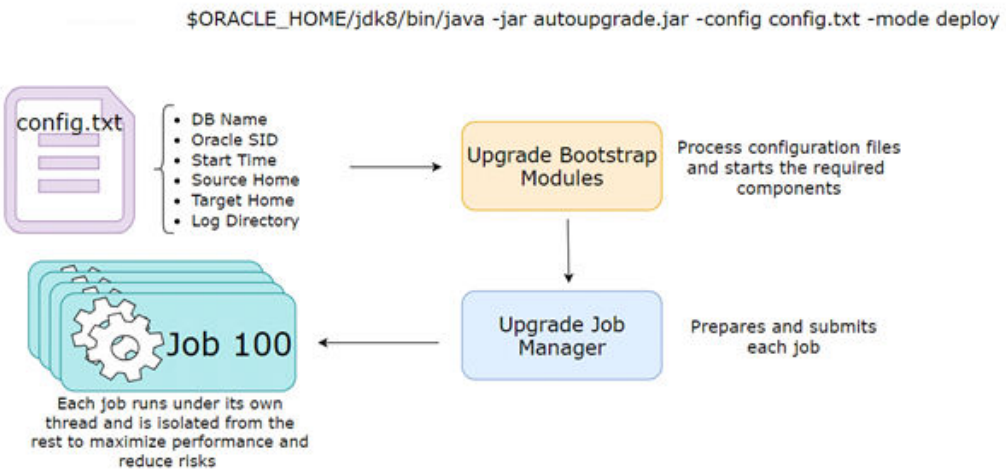
Understanding AutoUpgrade Workflows and Stages

The AutoUpgrade workflow automates each step of a typical upgrade process. The stages that run depend on the processing mode that you select.

AutoUpgrade is designed to enable you to perform an upgrade with as little human intervention as possible. When you start AutoUpgrade, the configuration file you identify with the command is passed to the AutoUpgrade job manager. The job manager creates the required jobs for the processing mode that you selected, and passes the data structures required for the mode to the dispatcher. The dispatcher then starts lower-level modules that perform each individual task.

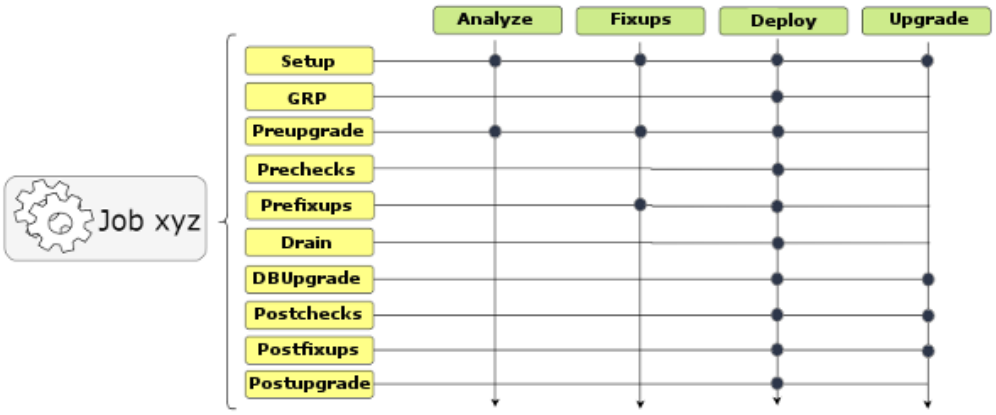
AutoUpgrade Processing Mode Workflow Processing

To understand how AutoUpgrade processes a workflow mode, review the following figure, which shows how a deploy processing mode is processed:



AutoUpgrade Processing Mode Stages

The stages that AutoUpgrade runs for an upgrade job depends on the processing mode that you select.



There are four AutoUpgrade modes. For each mode, AutoUpgrade steps are performed in sequence. Note the differences in steps for each mode

- **Analyze** Mode: Setup, Prechecks.
- **Fixups** Mode: Setup, Prechecks, and Prefixups.
- **Deploy** Mode: Setup, guaranteed restore point (GRP), Preupgrade, Prechecks, Prefixups, Drain, DB (database) Upgrade, Postchecks, and Postupgrade. You can run your own scripts before the upgrade (Preupgrade stage) or after the upgrade (Postupgrade stage), or both before and after the upgrade.
- **Upgrade** Mode: Setup, DB (database) Upgrade, Postchecks, and Postfixups.

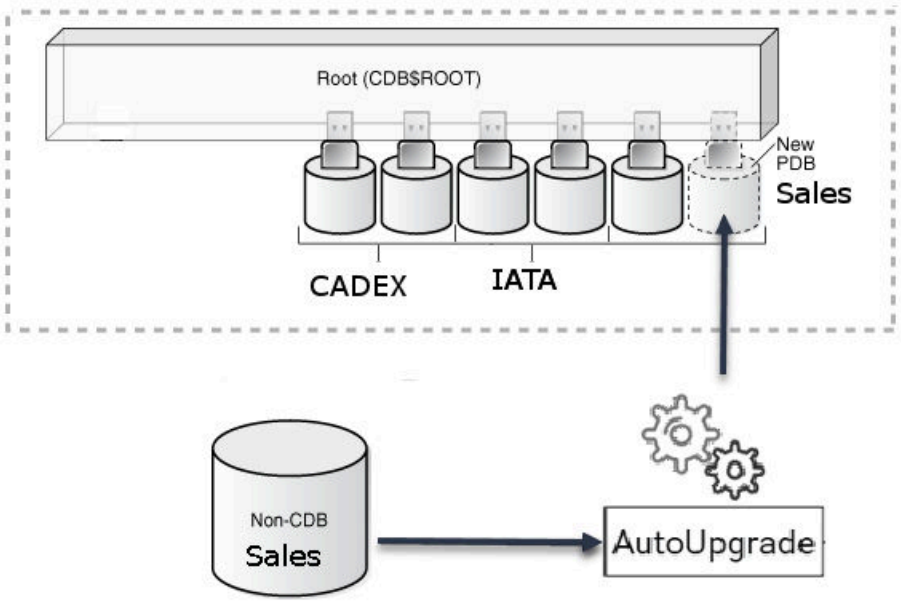
Understanding Non-CDB to PDB Upgrades with AutoUpgrade

You can upgrade and convert a non-CDB to a PDB in a new CDB in a single operation, or upgrade and then convert a Non-CDB database to a PDB in a pre-existing CDB.

Oracle Database 19c is the terminal release in which non-CDB Oracle Database architecture is supported. Oracle strongly recommends that you move to using pluggable databases (PDBs). When you migrate your database from the non-CDB architecture to PDBs in Oracle Database 19c, you obtain up to three user-configurable PDBs in a container database (CDB), without requiring a multitenant license. If you choose to configure four or more PDBs, then a multitenant license is required.

The non-CDB to PDB feature of the AutoUpgrade utility provides you flexible options to control how you upgrade your existing Oracle Database when you upgrade and convert an earlier release non-CDB architecture Oracle Database to a multitenant architecture database. You can perform this upgrade and conversion in a single operation.

Figure 3-1 Converting and Upgrading a Non-CDB Using AutoUpgrade



Example 3-1 AutoUpgrade Configuration File for Non-CDB to PDB Conversion

To use the non-CDB to PDB option, you must set the parameters `target_cdb` in the AutoUpgrade configuration file. The `target_cdb` parameter value defines the Oracle

system identifier (SID) of the container database into which you are plugging the non-CDB Oracle Database. For example:

```
global.autoupg_log_dir=/home/oracle/autoupg
upg1.sid=s12201
upg1.source_home=/u01/product/12.2.0/dbhome_1
upg1.log_dir=/home/oracle/autoupg
upg1.target_home=/u01/product/19.1.0/dbhome_1
upg1.target_base=/u01
upg1.target_version=19.1.0
upg1.target_cdb=cdb19x
```

Related Topics

- Permitted Features, Options, and Management Packs by Oracle Database Offering

AutoUpgrade Command-Line Parameters

Review the AutoUpgrade parameters and select the parameters and options for your Oracle Database upgrade use case.

Use the parameters with the command `java -jar autoupgrade.jar`.

- AutoUpgrade Command-Line Syntax
  To see how to use AutoUpgrade to perform your upgrades, review the syntax and run time use cases.
- Debug
  The AutoUpgrade parameter `debug` turns on the AutoUpgrade debug message feature, which assists you with correcting faulty AutoUpgrade job syntax.
- Clear_recovery_data
  The AutoUpgrade parameter `clear_recovery_data` removes the recovery information, which causes AutoUpgrade to start from the beginning on specified databases, or on all databases.
- Config
  The AutoUpgrade parameter `config` identifies the configuration file that you use to provide information about databases that you want to upgrade.
- Config_Values
  The AutoUpgrade parameter `config_values` enables you to provide the same input values about systems as a text configuration file. You can use it conjunction with the `config` parameter.
- Console
  The AutoUpgrade parameter `console` turns on the AutoUpgrade console, and provides a set of commands to monitor the progress of AutoUpgrade jobs.
- Create_sample_file
  The AutoUpgrade parameter `create_sample_file` generates either a configuration file, or a settings file. You edit these files to create production configuration or settings files for AutoUpgrade.
- Error_code
  The AutoUpgrade parameter `error_code` shows the error codes for AutoUpgrade errors.

- Mode
  The AutoUpgrade parameter `mode` value sets the mode from which AutoUpgrade runs.
- Noconsole
  The AutoUpgrade parameter `noconsole` turns off the AutoUpgrade console, so that AutoUpgrade runs using only configuration file information.
- Preupgrade
  The AutoUpgrade parameter `preupgrade` runs database checks and preupgrade fixups that fix most issues before you start an upgrade, and postupgrade fixups that fix most issues after an upgrade is completed.
- Settings
  The AutoUpgrade parameter `settings` identifies the configuration file that you use to provide custom runtime configuration of the AutoUpgrade utility.
- Version
  The AutoUpgrade parameter `version` prints to the terminal screen the current build of the `autoupgrade.jar` file.
- Restore
  The AutoUpgrade parameter `restore` performs a system-level restoration of the AutoUpgrade jobs that you specify.
- Restore_on_fail
  The AutoUpgrade parameter `restore_on_fail` automatically restores any job that failed during the deployment.
- Zip
  The AutoUpgrade parameter `zip` creates a zip file of log files required for filing an AutoUpgrade service request.

AutoUpgrade Command-Line Syntax

To see how to use AutoUpgrade to perform your upgrades, review the syntax and run time use cases.

Prerequisites

- You must have Java Development Kit (JDK) 8 or later installed in your source environment.

  JDK 8 is installed with every release starting with Oracle Database 12c Release 2 (12.2). For any release earlier than 12.2, you must either run AutoUpgrade using the Java release in the target Oracle Database, or you must install JDK 8 on your source database server.
- Oracle Database upgrades using the AutoUpgrade utility follow the same upgrade rules that apply to manual Oracle Database upgrades. Confirm that your source Oracle Database release is supported for upgrade.

With non-CDB to PDB conversion and upgrade, AutoUpgrade can automatically complete both upgrade and conversion when these conditions are met:

- The target release CDB must exist.
- In the AutoUpgrade configuration file, where the target CDB system identifier is *`target_cdb`*, you must set the local parameter `target_cdb` using the following syntax:

`target_cdb=target_cdb`. For example:

```
target_cdb=cdb1
```

- The `target_cdb` value is the Oracle SID of the CDB into which you are plugging the non-CDB.

File Path

The AutoUpgrade utility is a Java JAR file that is located in the new release Oracle Database home.

Oracle_home`/rdbms/admin/autoupgrade.jar`
Oracle strongly recommends that you obtain the latest AutoUpgrade JAR file from My Oracle Support. The JAR file and deployment instructions for the JAR file are available from My Oracle Support note 2485457.1

Syntax

AutoUpgrade command syntax is case-sensitive. Enter commands in lowercase.

```
java -jar autoupgrade.jar [options]
```

Multiple options can be concatenated.

Run Type One (Basic) Parameters for AutoUpgrade

Run type one (Basic) parameters and options for AutoUpgrade provide a starting point for preparing for upgrades.

Parameter	Description
`-version`	Displays the AutoUpgrade version.
`-help`	Displays the help file for AutoUpgrade syntax.
`-create_sample_file [settings | config` *config-file-name*`]`	Creates an example configuration file for AutoUpgrade. For a description of the options, see the `create_sample_file` parameter topic.

Run Type Two (Core) Parameters for AutoUpgrade

Run type two (Core) parameters and options for AutoUpgrade provide essential upgrade functionality for most upgrade scenarios.

Parameter	Description
`-config [`*config_path* `| -config_values` *config_values*`]`	Identifies the configuration file that you use to provide information about databases that you want to upgrade. For a description of the options, see the `config` parameter topic.
`-mode [analyze|fixups|deploy|upgrade|postfixups]`	Sets the mode from which AutoUpgrade runs. For a description of the options, see the `mode` parameter topic.
`-restore -jobs` *job#*	Performs a system-level restoration of the AutoUpgrade jobs that you specify

Parameter	Description
`-restore_on_fail`	If set, then when a job fails, the database is restored automatically. Errors in PDBs are not considered fatal, only errors in `CDB$ROOT` or Non-CDBs.
`-console`	Starts AutoUpgrade with the console enabled.
`-noconsole`	Starts AutoUpgrade with the console disabled.
`-debug`	Enables debug messages.
`-clear_recovery_data [-jobs` *`job#,job#,...`*`]`	Removes the recovery information, which causes AutoUpgrade to start from the beginning on all databases, or on databases in a comma-delimited list specified by `-jobs`. For a full description of the options, see the `clear_recovery_data` parameter topic.
`-restore -jobs` *`job#,job#,...`*	Runs a system-level restoration of the specified jobs. The databases are flashed back to the Guaranteed Restore Point (GRP). Before you run this command, the GRP must already be created by AutoUpgrade. For a full description of the options, see the `clear_recovery_data` parameter topic.
`-zip [-sid` *`sid`*`] [-d` *`dir`*`]`	Zips up log files required for filing an AutoUpgrade service request. For a description of the options, see the `zip` parameter topic.

Run Type Three (Additional) Parameters for AutoUpgrade

Run type three (Additional) parameters and options for AutoUpgrade are useful for particular upgrade scenarios, such as restarting from a failed point in an upgrade, or running particular fixups.

Parameter	Description
`-debug`	Enables debug messages.
`-error_code`	Displays the AutoUpgrade error codes.
`-help`	Displays the help file for AutoUpgrade syntax.
`-mode [analyze|fixups|postfixups]`	Sets the mode from which AutoUpgrade runs. For a description of the options, see the `mode` parameter topic.
`-preupgrade preupgrade_options` *`options`*	Runs database checks and preupgrade fixups that fix most issues before you start an upgrade, and postupgrade fixups that fix most issues after an upgrade is completed. For a description of the options, see the `preupgrade` parameter topic.

Related Topics

- My Oracle Support note 2485457.1
- Oracle Database Releases That Support Direct Upgrade

Debug

The AutoUpgrade parameter `debug` turns on the AutoUpgrade debug message feature, which assists you with correcting faulty AutoUpgrade job syntax.

Property	Description
Parameter type	string
Syntax	`autoupgrade.jar -parameter -debug`

Description

The AutoUpgrade `debug` parameter turns on debugging messages, which can assist you with correcting AutoUpgrade command syntax.

Usage Notes

Use the `debug` parameter in concert with any other AutoUpgrade parameter.

Clear_recovery_data

The AutoUpgrade parameter `clear_recovery_data` removes the recovery information, which causes AutoUpgrade to start from the beginning on specified databases, or on all databases.

Property	Description
Parameter type	string
Syntax	`clear_recovery_data [-jobs job_numbers]` where: *job_numbers* is a comma-delimited list of jobs that you want to clear

Description

The AutoUpgrade `clear_recovery_data` parameter removes the recovery information, which causes AutoUpgrade to start from the beginning on specified databases, or on all databases.

Usage Notes

Use after manually restoring a database and attempting a new upgrade. If no list of jobs is provided, then by default, all job metadata is removed. Removing the metadata does not remove log files, or reset the job identifier (`jobid`) counter. Only the AutoUpgrade files used to keep track of the progress of each job are removed.

Examples

The following example shows how to use the `clear_recovery_data` option after you encounter an issue, fix it, and then run AutoUpgrade again.

You start AutoUpgrade in deploy mode

```
java -jar autoupgrade.jar -config config.cfg -mode deploy
```

However, you encounter an issue during the upgrade. You stop AutoUpgrade, restore the database, and make changes to the database to correct the issue. To start over the AutoUpgrade procedure and clear out the current job state information, specify the job number that is associated with the previously run job. If you specify the job number, then AutoUpgrade only removes the state information for that specific job. The rest of the jobs will remain untouched.

```
java -jar autoupgrade.jar -config config.cfg -mode analyze -
clear_recovery_data -jobs 100
```

> **Note:**
>
> The job ID number associates the job with the database. If you enter the wrong job id, then that causes AutoUpgrade to restart the wrong job from the beginning.

The analyze results are good, so you then run the deploy option again:

```
java -jar autoupgrade.jar -config config.cfg -mode deploy
```

When you run `autoupgrade.jar -config` with the `-clear_recovery_data` parameter, AutoUpgrade only drops state files. It ignores any previously generated log files, so you can retain log files for further reference. Running AutoUpgrade with the `-clear_recovery_data` parameter also preserves the latest `jobid` information, so that the `jobid` AutoUpgrade creates for the next job is the next ID in sequence. By maintaining the `jobid` state, AutoUpgrade helps you to avoid mixing log output from earlier AutoUpgrade jobs in the same log file.
The following are additional examples of how you can run the `clear_recovery_data` parameter.

```
java -jar autoupgrade.jar -config config.cfg -clear_recovery_data
java -jar autoupgrade.jar -config config.cfg -clear_recovery_data -jobs 111,222
```

Config

The AutoUpgrade parameter `config` identifies the configuration file that you use to provide information about databases that you want to upgrade.

Property	Description
Parameter type	string
Syntax	`-config` *`configfile`*
Default value	None

Description

The `config` parameter specifies a configuration file name. It takes two arguments:

- The configuration file name
- (Optional) The path to the configuration file, as represented by *`config-file`*

Examples

Running AutoUpgrade with a configuration file named `myconfig.cfg`, with the processing mode `deploy`:

```
java -jar autoupgrade.jar -config myconfig.cfg -mode deploy
```

Config_Values

The AutoUpgrade parameter `config_values` enables you to provide the same input values about systems as a text configuration file. You can use it conjunction with the `config` parameter.

Property	Description
Parameter type	String.
Syntax	`-config_values [`*`config-parameter1=value*`*`,``config-parameter2=value,...]`
Default value	None.

Description

The `config_values` parameter enables you to provide values about database paths, instances, and target releases through the AutoUpgrade command line that otherwise require you to specify a configuration file. AutoUpgrade then creates a configuration file as the utility runs. Using `config_values` enables you to run AutoUpgrade without a configuration file.

The `config_values` options are a comma-delimited list that can support multiple database upgrades. Each database configuration is separated by asterisks (*) to identify different databases. Global entries must include the global prefix in the name. For example:

```
global.autoupg_log_dir=/u01/app/oracle/cfgtoollogs/upgradelogs/
```

Local entries only need to include the name:

```
target_home=/u01/app/oracle/product/21.0.0.0/dbhome_1
```

Logging directories are resolved in the following manner.

- Case: Global `autoupg_log_dir` is not specified.

If the `config_file` parameter is not passed to AutoUpgrade, then the local directory is used as the global log directory. If the `config_file` parameter is not passed to AutoUpgrade, then the global log directory defaults to the Java temporary directory:

  - Unix and Linux systems: `/tmp/autoupgrade`
  - Microsoft Windows: `C:\Users\name\AppData\Local\Temp\autoupgrade`
  - A configuration file is created with the name `autoupgradeYYYYMMMHHMMSS.cfg`, where *YYYY* is year, *MMM* is month, *HH* is hour, *MM* is minute, and *SS* is second.

- Case: `Global autoupg_log_dir` is specified.

  If the `config_file` parameter does not pass the directory to AutoUpgrade, then AutoUpgrade creates a configuration file in the AutoUpgrade log directory specified by the parameter. If the `config_file` parameter does not pass the directory to AutoUpgrade, then the configuration file is created under the global log directory. If you specify a configuration file name that already exists, then AutoUpgrade renames the existing configuration file using the suffix *YYYYMMMHHMMMSS*`.cfg`, where *YYYY* is year, *MMM* is month, *HH* is hour, *MM* is minute, and *SS* is second. For example: on April 29, 2020, at 08:30:04, if configuration file `\tmp\autoupgrade.cfg` already exists, and you pass the file name `-config_file \tmp\autoupgrade.cfg` to AutoUpgrade, then the existing file is renamed to `\tmp\autoupgrade.cfg20200429083004`. AutoUpgrade then creates the new configuration file `\tmp\autoupgrade.cfg`.

If you use the `-config_values` parameter, and the user account running the AutoUpgrade command has the following operating system environment variables set, then AutoUpgrade picks up the path defined for these variables:

- `ORACLE_HOME` - The Oracle home path for the source Oracle home
- `ORACLE_TARGET_HOME` - The target Oracle home path.
  - Linux and Unix: Equivalent to an `export ORACLE_TARGET_HOME` command. For example: `export ORACLE_TARGET_HOME=/u01/app/oracle/product/21.0.0/`
  - Microsoft Windows: Equivalent to a `SET ORACLE_TARGET_HOME` command. For example: `SET ORACLE_TARGET_HOME=C:\oracle\21`
- `ORACLE_SID` - The Oracle Database system identifier (SID).
  - Linux and Unix: Set with the operating system shell command `export ORACLE_SID`. For example: `export ORACLE_SID=sales`
  - Microsoft Windows: Set with the operating system shell command `SET ORACLE_SID` command. For example: `SET ORACLE_SID=sales`
- `ORACLE_TARGET_VERSION` - The target release of the new Oracle home. You must set this operating system environment variable either when the target Oracle home does not exist, or the target home is a release earlier than Oracle Database 18c.
  - Linux and Unix: Set with `export ORACLE_TARGET_VERSION`. For example, for Oracle Database 19c:

    ```
    export ORACLE_TARGET_VERSION=19.1
    ```

For Oracle Database 21c:

```
export ORACLE_TARGET_VERSION=21.1
```

– Microsoft Windows: Set with `SET ORACLE_TARGET_VERSION`.

For example, for Oracle Database 19c:

```
SET ORACLE_TARGET_VERSION=19.1
```

For example, for Oracle Database 21c:

```
SET ORACLE_TARGET_VERSION=21.1
```

If you use the `config_values` parameter in place of a configuration file, and you do not have these operating system environment variables set for the user account running AutoUpgrade, then you must provide at least these four values as arguments using `config_values`.

Example: Running AutoUpgrade With an Existing Configuration File

Scenario: Running AutoUpgrade with an existing configuration file, using `config_values`. The following command syntax creates the `global.autoupg_log_dir` from the local directory where the `myconfig.cfg` file is created. As a result of this command, the location for `global.autoupg_log_dir` is set to `/dir`:

```
java –jar autoupgrade.jar –config /dir/myconfig.cfg –config_values
        “source_home=/srcdir, target_home=/trgdir, sid=sales” –mode
        deploy
```

The configuration file `myconfig` is created in the path `/dir`, with the following entries:

```
global.autoupg_log_dir=/dir
autoupgrade1.source_home=/srcdir
autoupgrade1.target_home=/trgdir
autoupgrade1.sid=sales
```

Example: Running AutoUpgrade Without Specifying a Value for `–config_values`

In analyze, fixup, upgrade, or deploy mode, if you have set user environment values that AutoUpgrade requires to run, and you do not pass these values as an argument for `–config_values`, then AutoUpgrade defaults to using the user environmental variables set on the server.

To understand how this works, suppose you run AutoUpgrade as the user `oracle`, for which the following environment variables are set, where the target version is Oracle Database 21c:

- `ORACLE_HOME` is set to `/u01/app/oracle/product/12.2.0.1/dbhome_1`
- `ORACLE_TARGET_HOME` is set to `/u01/app/oracle/product/21.0.0.0/dbhome_1`
- `ORACLE_SID` is set to `sales`
- `ORACLE_TARGET_VERSION` is set to `21.1`

Now suppose you run the following command at 11:45:15 AM on September 30, 2020:

```
[Wed Sep 30 11:45:15] oracle@example:~$ java –jar autoupgrade.jar –
config_values –mode analyze
```

Because the log directory was unspecified, AutoUpgrade defaults writing the configuration file for the run to the temporary directory. The configuration file AutoUpgrade creates resides in the path `/tmp/autoupgrade` as the file`/tmp/autoupgrade/autoupgrade20200501114515.cfg`, with the following entries:

```
global.autoupg_log_dir=/tmp/autoupgrade
# Value from environmental variable ORACLE_HOME
autoupgrade1.source_home=/u02/app/oracle/122
# Value from environmental variable ORACLE_TARGET_HOME
autoupgrade1.target_home=/scratch/oracle/21
# Value from environmental variable ORACLE_SID
autoupgrade1.sid=sales
# Value from environmental variable ORACLE_TARGET_VERSION
autoupgrade1.target_version=21.3
```

This option enables you to use AutoUpgrade to handle a single database upgrade without requiring you to specify extensive details about the upgrade.

Example: Running AutoUpgrade with `–config_values` entries for multiple databases

In this scenario, you run AutoUpgrade with `–config_values` entries for multiple databases, using * to delimit values for each database, with a target release of Oracle Database 21c:

```
 java –jar autoupgrade.jar –config /tmp/auto.cfg
–config_values "global.autoupg_log_dir=/scratch/upglogs,source_home=/
scratch/122,target_home=/scratch/21,sid=sales,*,source_home=/scratch/
18,target_home=/scratch/21,sid=employees"
```

The configuration file is created in the directory `/tmp` as `/tmp/auto.cfg`, with the following entries.

```
global.autoupg_log_dir=/scratch/upglogs
autoupgrade1.source_home=/scratch/122
autoupgrade1.target_home=/scratch/21
autoupgrade1.sid=sales
autoupgrade2.source_home=/scratch/18
autoupgrade2.target_home=/scratch/21
autoupgrade2.sid=employees
```

Console

The AutoUpgrade parameter `console` turns on the AutoUpgrade console, and provides a set of commands to monitor the progress of AutoUpgrade jobs.

Property	Description
Parameter type	string
Syntax	`autoupgrade.jar -config your-file -mode your-mode`

Description

To monitor upgrades, use the AutoUpgrade parameter `console` to run the Console, which monitors the status of upgrade jobs.

The AutoUpgrade console starts by default with the AutoUpgrade command. You can reenable or disable the AutoUpgrade console using the option `-console|-noconsole`

When you use the `-noconsole` option, AutoUpgrade runs using only the settings in the configuration file, without requiring console input. Use the `noconsole` option when you want to create scripts for AutoUpgrade, such as in cases where you want to analyze multiple databases. After the AutoUpgrade jobs are finished, you can review the output of the Analyze mode logs to see what is required to upgrade each of the databases included with your configuration script.

> **Note:**
>
> You can start as many instances of AutoUpgrade as you want, but each instance must use a unique global logging directory (`global.autoupg_log_dir`). If you only have one global logging directory, then you can only start one instance.

Usage Notes

When you start the console, you can use options within the console.

Console Option	Description
`-exit`	Closes and exits from the console. If there are jobs running, then they are stopped.
`-help`	Displays the console command help.
`-lsj [(-r|-f-p|-e)|-n number]`	Lists jobs by status, up to the number of jobs you specify with the numeric value *number*. You can use the following flags: `-f`: (Optional) Filter by finished jobs. `-r`: (Optional) Filter by running jobs. `-e`: (Optional) Filter by jobs with errors. `-p`: (Optional) Filter by jobs in preparation. `-n number`: (Required) Number of jobs to display, specified by integer value.
`-lsr`	Displays the restoration queue.
`-lsa`	Displays the abort queue.
`-tasks`	Displays the tasks that are running.

Console Option	Description
`-clear`	Clears the terminal display.
`-resume -job number`	Restarts from a previous job that was running, specified by a numeric value (*number*) for the job.
`-status [-job number|-long]`	Lists the status of a particular job, specified by a numeric value (`-job number`). When run with `-long`, AutoUpgrade displays full details about the job as it is being run.
`-restore [-job (0-9)|-all_failed]`	Restores the database in the AutoUpgrade job specified by the integer value *number* to its state before starting the upgrade. When run with the `all_failed` option, restores all failed jobs to their previous state before the upgrade started.
`-logs`	Displays all log file locations.
`-abort -job number`	Aborts the job specified by the numeric value that you provide (*number*).
`-h[ist][/number]`	Displays the console command-line history, and takes the option to run a command again, depending on the flat with which you run it: Flags: `/` Runs the last command again. `/ number` Runs the command in the history log specified by the command line number that you specify.

Create_sample_file

The AutoUpgrade parameter `create_sample_file` generates either a configuration file, or a settings file. You edit these files to create production configuration or settings files for AutoUpgrade.

Property	Description
Parameter type	string
Syntax	`-create_sample_file config [filename] [type =[full|unplug|noncdbtopdb]] | settings [filename]`

Property	Description
Default value	For `create_sample_file config`, if you append a filename to the command, then an example configuration file is created with the name you provide. If you do not provide an output file name, then the example configuration file is created with the name `sample_config.cfg`. When you add the `type =` clause, you select one of the following options: • `full` : A complete options AutoUpgrade configuration file • `unplug` : An AutoUpgrade configuration file with options for unplug-plug upgrades of PDBs. • `noncdbtopdb` : An AutoUpgrade configuration file with options for nonCDB to PDB upgrades. When you add the `settings` clause, an internal settings configuration file is generated. You can accept the default file name, or specify a file name.

Usage Notes

The `create_sample_file` parameter is optional. It cannot be used together with other parameters. When you specify this parameter, it requires either the `settings` or the `config` clause:

`settings`: Generates an AutoUpgrade **settings** file, either with the name `sample_autoupg.cfg`, or with a name that you specify.

`config`: Generates an AutoUpgrade **configuration** file, either with the name `sample_config.cfg`, or with a name that you specify.

After you generate one of these example files, you can modify the file to control how the AutoUpgrade utility performs upgrades.

- `config`: Generates a template upgrade configuration file of a configuration mode type. AutoUpgrade generates a file named `sample_config.cfg`, or with a name you provide, in the current folder
- `settings` AutoUpgrade generates a file named `sample_autoupg.cfg`, or with the name you provide, in the current folder.

For both the `config` and `settings` options, the default file name is generated with the extension `.cfg`. However, AutoUpgrade can read files without an extension, or with an extension that you provide, as long as the file is a valid (plain text) file. The default extension is for convenience in identifying these files as configuration files.

Generating an example configuration file is a standard part of preparing to use AutoUpgrade. After you customize the configuration file parameters in the example configuration file, you can use that file as the production settings and configuration file for your upgrade.

> **Caution:**
>
> The settings file is used to overwrite internal settings of the AutoUpgrade. Generating an example settings file is not required for most use cases. Use carefully.

Examples

Example of running the `create_sample_file` parameter with the `config` clause:

```
[oracle@example ~]$ java -jar autoupgrade.jar -create_sample_file config
Created sample configuration file /home/oracle/sample_config.cfg
```

Example of running the `create_sample_file` parameter with the `config`, clause specifying an output configuration file name:

```
[oracle@example ~]$ java -jar autoupgrade.jar -create_sample_file
config sales01
Created sample configuration file /home/oracle/sales01.cfg
```

Example of running the `create_sample_file` parameter with the `settings` clause:

```
oracle@example ~]$ java -jar autoupgrade.jar -create_sample_file
settings
Created sample settings file /home/oracle/sample_autoupg.cfg
```

Example of running the `create_sample_file` parameter with the `settings`, clause specifying an output configuration file name:

```
oracle@example ~]$ java -jar autoupgrade.jar -create_sample_file
settings testsetting.test
Created sample settings file /home/oracle/testsetting.test
```

Error_code

The AutoUpgrade parameter `error_code` shows the error codes for AutoUpgrade errors.

Property	Description
Parameter type	string
Syntax	`-error_code [`*`errorcode`*`]`
Default value	When no error code is specified, all AutoUpgrade error codes are displayed in the Console. When an error code is specified, the information about the specified error code is displayed in the Console.

Examples

When entered without a specification, Autoupgrade produces descriptions of all of the error codes:

```
$ java -jar autoupgrade.jar  -error_code
ERROR1000.ERROR  = UPG-1000
ERROR1000.CAUSE  = It was not possible to create the data file where the
jobsTable is being written or there was a problem during the writing, it might
be thrown due to a permission error or a busy resource scenario

ERROR1001.ERROR  = UPG-1001
ERROR1001.CAUSE  = There was a problem reading the state file perhaps there was
corruption writing the file and in the next write it might be fixed

ERROR1002.ERROR  = UPG-1002
ERROR1002.CAUSE  = Error deserializing the object for rerun, review log for any
errors
.
.
.
```

When entered with a specific error code, AutoUpgrade provides output for the error that you specify. For example:

```
java -jar autoupgrade.jar -error_code UPG-3010
```

This command produces the following output:

```
ERROR3010.ERROR  = UPG-3010
ERROR3010.CAUSE  = Error running approot_to_pdb.sql script
```

Here is another example:

```
$ java -jar autoupgrade.jar  -error_code UPG-1400
```

This command produces the following output:

```
ERROR1400.ERROR  = UPG-1400
ERROR1400.CAUSE  = Database upgrade failed with errors
```

Mode

The AutoUpgrade parameter `mode` value sets the mode from which AutoUpgrade runs.

Property	Description
Parameter type	string
Syntax	`-mode = [analyze|fixups|deploy|upgrade|postfixups]`

Property	Description
Default value	None. Choose one of the following options: • `analyze` : Runs upgrade readiness checks in the source Oracle home. • `fixups` : Runs the upgrade readiness checks and preupgrade fixups, but does not perform the upgrade. • `deploy` : Performs the upgrade of the databases from start to finish. • `upgrade` : Performs the database upgrade and postupgrade actions. Databases in the target Oracle homes must be up and running before you start this mode. • `postfixups` Runs postfixups of databases in the target Oracle home.

Examples

```
java -jar autoupgrade.jar -config config.cfg -mode analyze
java -jar autoupgrade.jar -config config.cfg -mode deploy
java -jar autoupgrade.jar -preupgrade "target_version=21" -mode fixups
```

Noconsole

The AutoUpgrade parameter `noconsole` turns off the AutoUpgrade console, so that AutoUpgrade runs using only configuration file information.

Property	Description
Parameter type	string
Syntax	`-noconsole`

Description

When you use the `noconsole` option, AutoUpgrade runs using only the settings in the configuration file, without requiring console input. Use the `noconsole` option when you want to run AutoUpgrade as part of a batch flow, or in scripts, such as in cases where you want to analyze multiple databases. After the AutoUpgrade jobs are finished, you can review the output of the Analyze mode logs to see what is required to upgrade each of the databases included with your configuration script.

> **Note:**
>
> You can run only one AutoUpgrade instance at a time that is associated with a given configuration file.

Usage Notes

In this example, AutoUpgrade is run in Analyze mode, using the configuration file in `noconsole` mode.

```
java -jar autoupgrade.jar -config autoupgrade.cfg -mode analyze -
noconsole
```

Preupgrade

The AutoUpgrade parameter `preupgrade` runs database checks and preupgrade fixups that fix most issues before you start an upgrade, and postupgrade fixups that fix most issues after an upgrade is completed.

Property	Description
Parameter type	string
Syntax	`-preupgrade preupgrade_options -mode [analyze|fixups|postfixups]`
Default value	`analyze`

Description

The -preupgrade clause of AutoUpgrade replaces the functions previously preformed by the manual Pre-Upgrade Information Tool (preupgrade.jar) in previous releases. The `-mode` clause takes one of three values:

- `analyze`: Check your system for readiness to upgrade.
- `fixups`: Perform fixups as needed on your source Oracle Database release in preparation for upgrade
- `postfixups`: Perform fixups on your target Oracle Database release after upgrade is completed.

If no value for `-mode` is specified, then by default the `-preupgrade` parameter defaults to `analyze` mode.

Usage Notes

Use the `preupgrade` clause only if you want to obtain the same features previously made available with the Pre-Upgrade Information Tool (`preupgrade.jar`). For most upgrade scenarios, you do not need to use this parameter.

The `-preupgrade` parameter requires `preupgrade_options`, which specifies a list of comma-delimited option-value pairs in the following format: *`option1=value1,option2=value2,…`*

Arguments

- `target_version=`*`release-number`*: Specifies the target Oracle Database release version, which is the release to which you want to upgrade.

The value for this argument is required by the `analyze` and `fixups` modes. However, the target release can be derived from `target_home`. Accordingly, for `analyze` and `fixups` modes, either `target_version` or `target_home` must be specified. The value for `target_version` must be `12.2`, or a later release value.

- `target_home=[`*`target-path`*`|`*`env-variable`*`]`: Specifies the Oracle Database home location of the target release to which you want to upgrade, which can either be the Oracle home path, or an operating system path variable.

  This argument is mandatory if you select the `postfixups` mode. If you select the `postfixups` mode, and you do not specify a target home path, then the default value is specified by the Oracle home environment variable for the Oracle home set for the user running AutoUpgrade (`$ORACLE_HOME` on Linux and Unix systems, or `%ORACLE_HOME%` on Microsoft Windows systems).

- `oh=[`*`source-path`*`|`*`env-variable`*`]`: Specifies the Oracle Database home location of the source release from which you want to upgrade, which can either be the Oracle home path, or an operating system path variable.

  This argument is mandatory if you select the `analyze` or `fixups` mode. If you select either `analyze` or `fixups` modes, and you do not specify a source home path, then the default value is specified by the Oracle home environment variable for the Oracle home set for the user running AutoUpgrade (`$ORACLE_HOME` on Linux and Unix systems, `%ORACLE_HOME%` on Microsoft Windows systems).

- `sid=`*`system-identifier`*: Specifies an Oracle system identifier for the source database that you want to upgrade. This argument is mandatory for `analyze` or `fixups` modes. If you select either the `analyze` or the `fixups` mode, and you do not specify a system identifier, then the default value is specified by the Oracle home environment variable for the Oracle home set for the user running AutoUpgrade (`$ORACLE_SID` on Linux and Unix systems, `%ORACLE_SID%` on Microsoft Windows systems).

- `dir=`*`path`*: Directs the output to a specific directory. If you do not specify an output directory with the `dir` argument, then the output is directed to a folder called `autoupgrade` that is placed in the temporary directory on your system. Typically, that directory is in one of the following locations:

  - Linux or Unix: `/tmp`, or `/var/tmp`.
  - Microsoft Windows: `C:\WINNT\TEMP`

- `inclusion_list=`*`list`*: Specifies a list of pluggable databases (PDBs) inside a container database (CDBs) that you want to include for processing. Provide a space-delimited list of PDBs that you want processed, in one of the following two formats, where `pdb1`, `pdb2`, and `pdb3` are PDBs that you want processed:

  - `pdb1 pdb2 pdb3`
  - `(pdb1 pdb2 pdb3)`

  If you do not specify a list of PDBs, then all PDBs in a CDB are processed.

- `exclusion_list=`*`list`*: Specifies a list of pluggable databases (PDBs) inside a container database (CDBs) that you want to exclude for processing. Provide a space-delimited list of PDBs that you want processed, in one of the following two formats, where `pdb1`, `pdb2`, and `pdb3` are PDBs that you want processed:

  - `pdb1 pdb2 pdb3`
  - `(pdb1 pdb2 pdb3)`

If you do not specify a list of PDBs, then all PDBs in a CDB are processed.

- `user=username`: Specifies the Oracle Database user name that the AutoUpgrade utility uses to connect to Oracle Database If the user is specified, then AutoUpgrade prompts for the user name password input on the command line. If no user name is specified, then AutoUpgrade uses operating system authentication for the Oracle Database connection.

Modes

- `analyze` (Default value): Runs Autoupgrade in Analyze mode, with all of the preupgrade checks that apply for the target release argument that you specify. If you do not specify a mode, then AutoUpgrade defaults to `analyze`.
- `fixups`: Runs preupgrade fixups (when available) for all issues reported by AutoUpgrade Analyze preupgrade checks on the source database that must be fixed before upgrade. All checks are run.

  Fixup results are reported in the file `upgrade.xml`. That file is placed in the path *`log_dir`*`/`*`db_name`*`/`*`jobnumber`*`/prefixups/prefixups.xml`, where *`log_dir`* is the log directory that you specify using the `dir` argument, *`db_name`* is the name of the source database, and *`jobnumber`* is the autoupgrade job number.
- `postfixups`: Runs postupgrade fixups (when available) for all issues reported by AutoUpgrade Analyze preupgrade checks on the upgraded database that you must fix after the upgrade is completed.

  Postfixup results are reported in the file `postfixups.xml`. That file is placed in the path *`log_dir`*`/`*`db_name`*`/`*`jobnumber`*`/postfixups`, where *`log_dir`* is the log directory that you specify using the `dir` argument, *`db_name`* is the name of the source database, and *`jobnumber`* is the autoupgrade job number.

Examples

Running AutoUpgrade with the preupgrade clause using `analyze` mode, and specifying that the target release is Oracle Database 21c.

```
java -jar autoupgrade.jar -preupgrade "target_version=21" -mode analyze
```

Running AutoUpgrade with the preupgrade clause using `fixups` mode, and specifying that the target release is Oracle Database 21c.

```
java -jar autoupgrade.jar -preupgrade "target_version=21" -mode fixups
```

Running AutoUpgrade with the preupgrade clause using `postfixups` mode, and specifying that the target Oracle home is in the path `C:\app\oracle\product\21.0.0\dbhome_1`.

```
java -jar autoupgrade.jar -preupgrade
"target_home=C:\app\oracle\product\21.0.0\dbhome_1" -mode postfixups
```

Running AutoUpgrade with the preupgrade clause without specifying the mode, and specifying that the target release is Oracle Database 21c. In this case, the mode used is the default mode, `analyze`.

```
java -jar autoupgrade.jar -preupgrade "target_version=21"
```

Settings

The AutoUpgrade parameter `settings` identifies the configuration file that you use to provide custom runtime configuration of the AutoUpgrade utility.

Property	Description
Parameter type	String
Syntax	`-settings` *`my-custom-advanced-settings`*
Default value	Not applicable

Description

The `settings` parameter has the required argument of the name and path to the settings configuration file, which you have modified with custom settings. The `settings` parameter cannot be used alone, but rather as a configuration input file that modifies the way that AutoUpgrade runs a processing mode.

Usage Notes

This parameter is an advanced parameter. For most upgrade scenarios, you should not need to modify any internal AutoUpgrade parameter settings.

Example

In this example, settings specifies a settings input file called my_custom_advanced_settings.cfg.

```
java -jar autoupgrade.jar -settings my_custom_advanced_settings.cfg -
config config.cfg -mode deploy
```

Version

The AutoUpgrade parameter `version` prints to the terminal screen the current build of the `autoupgrade.jar` file.

Property	Description
Parameter type	string
Syntax	-version
Default value	Not applicable.

Description

Use this optional parameter to check which version of the autoupgrade.jar utility is on your server.

Usage Notes

Command Example:

```
java -jar autoupgrade.jar -version
```

Output example:

```
[oracle@example ~]$ java -jar autoupgrade.jar -version

build.hash 04a58df
build.version 20201127
build.date 2020/11/27 11:40:37
build.max_target_version 20
build.type production-21.1
```

Restore

The AutoUpgrade parameter `restore` performs a system-level restoration of the AutoUpgrade jobs that you specify.

Property	Description
Parameter type	string
Syntax	`-restore -jobs` *`job#,job#`*
Default value	Not applicable.

Description

Use this optional parameter to specify a system-level restoration of the jobs you specify, using a comma-delimited list of job numbers. The databases in the upgrade jobs that you specify are flashed back to the Guarantee Restore Point (GRP). Before you run this command, the GRP must have been created by AutoUpgrade.

Examples

```
java -jar autoupgrade.jar -config config.cfg -restore -jobs 111
java -jar autoupgrade.jar -config config.cfg -restore -jobs 111,222 -console
java -jar autoupgrade.jar -config config.cfg -restore -jobs 111,222 -noconsole
```

Restore_on_fail

The AutoUpgrade parameter `restore_on_fail` automatically restores any job that failed during the deployment.

Property	Description
Parameter type	string
Syntax	`-restore_on_fail`
Default value	Not applicable.

Description

Use this optional parameter to specify that AutoUpgrade restores any jobs that failed during the upgrade deployment.

Examples

```
java -jar autoupgrade.jar -config config.cfg -mode deploy -restore_on_fail
```

Zip

The AutoUpgrade parameter `zip` creates a zip file of log files required for filing an AutoUpgrade service request.

Property	Description
Parameter type	string
Syntax	`-zip [-sid sid] [-d dir]`
Default value	Not applicable.

Description

Use this optional parameter to create a zip file that you can send to Oracle Support that contains the log files for jobs that are the object of your service request. Use the `-sid` clause to specify a comma-delimited list of system identifiers (SIDs) of databases whose log files you want to send. If no SID value is defined, then AutoUpgrade creates a zip file for all databases specified in the configuration file. Use the `-d` clause to specify a specific output directory. If no directory is specified, then the current directory from which the command is run is used for the zip file output.

Usage Notes

> **Note:**
>
> When you use the `-zip` clause, you cannot use the `-mode` clause.

Examples

```
java -jar autoupgrade.jar -config yourconfig.cfg -zip
java -jar autoupgrade.jar -config yourconfig.cfg -zip -sid sales1,sales2 -d /
scratch/upgrd
```

AutoUpgrade Utility Configuration Files

AutoUpgrade configuration files contain all the information required to perform Oracle Database upgrades.

AutoUpgrade configuration files contain **global** and **local** configuration parameters. Global parameters by default apply to all databases addressed by the configuration file. When specified for a specific database, local configuration parameters override global parameters specified by the configuration file.

- Global Parameters for the AutoUpgrade User Configuration File
  To specify a default behavior for a parameter for all Oracle Database upgrades addressed in the configuration file, you can use the optional AutoUpgrade global parameters.
- Local Parameters for the AutoUpgrade Configuration File
  To configure information for specific Oracle Databases for the AutoUpgrade utility upgrade, you provide information in the AutoUpgrade local parameters.
- Locally Modifiable Global Parameters for AutoUpgrade Configuration File
  Locally modifiable global parameters are parameters that you set both globally, and as you require, set locally, so that you can better control AutoUpgrade job processing.

Global Parameters for the AutoUpgrade User Configuration File

To specify a default behavior for a parameter for all Oracle Database upgrades addressed in the configuration file, you can use the optional AutoUpgrade global parameters.

Usage Notes

All global parameters are optional. All global parameters take the prefix `global`.

The `add_after_upgrade_pfile` and `del_during_upgrade_pfile` global and local `PFILE` parameters operations are run in the following hierarchical order:

1. Global Actions
   a. Remove global
   b. Add global
2. Local Actions
   a. Remove local
   b. Add local

Table 3-1 Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`add_after_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to add after the `PFILE` is upgraded. This specification applies to all databases in the user configuration file. Example: `global.add_after_upgrade_pfile=/path/to/my/add_after.ora`
`add_during_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to have added during the `PFILE` upgrade. This specification applies to all databases in the user configuration file. `global.add_during_upgrade_pfile=/path/to/my/add_during.ora`

Table 3-1 (Cont.) Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`after_action`	(Optional) Specifies a path and a file name for a custom user script that you want to have run after all the upgrade jobs finish successfully. The script that you use must be in the form of *`name.ext`* (for example, `myscript.sh`, so that AutoUpgrade can identify the type of script that you want to run. Permitted extension options:
	• Unix shell (`.sh`)
	• Microsoft Windows batch (`.bat`, `.cmd`)
	• Microsoft Windows PowerShell (`.ps1`)
	By default, if the script fails, then AutoUpgrade continues to run. Use the `Y` flag to specify that AutoUpgrade stops if the operating system detects that your script fails. If the script finishes with a status different than `0`, then it is considered a failed completion.
	Examples:
	If the script fails, then stop AutoUpgrade:
	`global.after_action=/path/to/my/script.sh Y`
	If the script fails, then continue AutoUpgrade:
	`global.after_action=/path/to/my/script.sh`

Table 3-1 (Cont.) Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`autoupg_log_dir`	(Optional) Sets the location of the log files, and temporary files that belong to global modules, which AutoUpgrade uses. Example: `global.autoupg_log_dir=`*`/path/to/my/global/log/dir`* Starting with AutoUpgrade 19.7, you can configure different log directory path in the `userconfig` file in the logs directory for a specific prefix `global.autoupg_log_dir=`*`/path/to/my/global/log/dir`* *`myprefix`*`.log_dir=global.auto_log_dir:`*`different/path`* The result of using this syntax is that log files and temporary files are placed in the following path for databases identified by the prefix *`myprefix`*: `/path/to/my/global/log/dir/different/path` If you do not set this parameter to a path, then by default the log files are placed in the location indicated by the `orabase` utility for the databases that you include in your configuration file. In that case, the default logs directory is in the path *`ORACLE_BASE`*`/cfgtoollogs/autoupgrade`. If the `orabase` utility fails for all databases included in the configuration file, then the log file location is then based on the `temp` directory for the user running AutoUpgrade.

Table 3-1 (Cont.) Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`before_action`	(Optional) Specifies a custom user script that you want to have run for all upgrades before starting the upgrade jobs. The script that you use must be in the form of *`name.ext`* (for example, `myscript.sh`), so that AutoUpgrade can identify the type of script that you want to run. If you want to have a script run before a specific upgrade job, then specify that script by using the local parameter (`local.before_action`)
	Permitted extension options:
	• Unix shell (`.sh`)
	• Microsoft Windows batch (`.bat`, `.cmd`)
	• Microsoft Windows PowerShell (`.ps1`)
	By default, if the script fails, then AutoUpgrade continues to run. Use the `Y` flag to specify that AutoUpgrade stops if the operating system detects that your script fails. If the script finishes with a status different than `0`, then it is considered a failed completion.
	Examples:
	If the script fails, then stop AutoUpgrade:
	`global.before_action=/path/to/my/script.sh Y`
	If the script fails, then continue AutoUpgrade:
	`global.before_action=/path/to/my/script.sh`

`catctl_options`	(Optional) Specifies one or more of a set of `catctl.pl` options that you can select for AutoUpgrade to submit for `catctl.pl` to override default behavior. For a complete description of the options, refer to "Parallel Upgrade Utility (`catctl.pl`) Parameters," which is linked to at the end of this table. Available `catctl.pl` options: • `-n` Number of processes to use for parallel operations. • `-N` Number of SQL processors to use when upgrading PDBs. • `-t` Run SQL in classic upgrade overwriting default replay upgrade method • `-T` Takes offline user schema-based table spaces. • `-z` Turns on production debugging information for `catcon.pm`. Example: `global.catctl_options=-t -n 24 -N 4`

Table 3-1 (Cont.) Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`del_after_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to have removed after the `PFILE` upgrade. This specification applies to all databases in the user configuration file. Example: `global.del_after_upgrade_pfile=/path/to/my/del_after.ora`
`del_during_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to have removed during the `PFILE` upgrade. This specification applies to all databases in the user configuration file. Example: `global.del_during_upgrade_pfile=/path/to/my/del_during.ora`

`drop_grp_after_upgrade`	(Optional) Deletes the Guaranteed Restore Point (GRP) after database upgrade. If you select this option, then GRP is deleted after upgrade completes successfully. If you set `raise_compatible` to `yes`, then you must also set the parameter `drop_grp_after_upgrade` to `yes`.
	Options:
	`[yes | no]`
	The default value is `no`.
	Example:
	`global.drop_grp_after_upgrade=yes`

`target_base`	(Optional) Specifies the target `ORACLE_BASE` path for the target Oracle home. Use of this parameter is only required in rare cases.
	Example:
	`global.target_base=/u01/app/oracle`
	`sales4.target_base=/u04/app/oracle4`

Table 3-1 (Cont.) Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`raise_compatible`	(Optional) Increases the compatible parameter to the default value of the target release after the upgrade is completed successfully.
	Options:
	`[yes | no]`
	The default value is `no`.
	CAUTION:
	• After the `COMPATIBLE` parameter is increased, database downgrade is not possible.
	• Oracle recommends that you only raise the `COMPATIBLE` parameter to the current release level after you have thoroughly tested the upgraded database.
	• Regardless of what value you use for the `autoupgrade` command-line parameter `restore`, if you set the value of the configuration file parameter `raise_compatible` to `yes`, then before starting the upgrade, you must delete manually any guaranteed restore point you have created. After the upgrade is completed successfully, AutoUpgrade deletes the guaranteed restore point it creates before starting the upgrade. When AutoUpgrade starts the POSTUPGRADE stage, there is no way to restore the database.
	• If you set `raise_compatible` to `yes`, then you must also set the parameter `drop_grp_after_upgrade` to `yes`.
	Example:
	`global.raise_compatible=yes`

`target_home`	(Optional for analyze and fixups modes. Required for upgrade and deploy modes.) Sets a global target home for all of the databases specified in the configuration file. Use this option to avoid specifying the same `target_home` multiple times. This parameter can be overwritten locally. Example: `global.target_home=/target/Oracle/home`

Table 3-1 (Cont.) Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`target_version`	(Optional) Specifies the target release version on which you want AutoUpgrade to perform the upgrade. AutoUpgrade uses the release version information that you provide in this parameter to ensure that the correct checks and fixups are used for the target Oracle Database release to which you are upgrading. The format for this parameter are period-delimited values of valid Oracle versions.
	Valid values
	• 12.2
	• 18
	• 19
	This option is only required if the target home is not present on the system, or if the target home is a 12.2 release. Otherwise, AutoUpgrade can derive the target release value.
	Example:
	`global.target_version=18`
	`employees.target_version=12.2`

`upgradexml`	(Optional) Generates the `upgrade.xml` file. Options: `[yes | no]` The `upgrade.xml` is equivalent to the file in earlier releases that the preupgrade package generated when you specified the XML parameter. This file is created during the analyze mode (`mode -analyze`). It is generated in the prechecks directory defined for the AutoUpgrade log files. Example: `global.upgradexml=yes`

Related Topics

- Parallel Upgrade Utility (catctl.pl) Parameters

Local Parameters for the AutoUpgrade Configuration File

To configure information for specific Oracle Databases for the AutoUpgrade utility upgrade, you provide information in the AutoUpgrade local parameters.

Usage Notes

Local parameters take precedence over any global parameters set in the AutoUpgrade configuration file. Except where indicated with (Optional), all local parameters are required. All local parameters take a prefix (in examples, identified by a value you define to identify a particular database or upgrade. The prefix identifies the specific upgrade job to which the parameter applies in the configuration file.

Example: The set of parameters for the first upgrade in the configuration file uses the prefix *sales*, and the set of parameters for the next upgrade in the configuration file uses the prefix *employees*:

```
sales.source_home=/u01/app/oracle/12.2/dbhome1
.
.
.
employees.sid=salescdb
employees.source_home-/03/app/oracle/21/dbhome1
```

Table 3-2 Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`add_after_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to add after the upgrade. Example: `sales3.add_after_upgrade_pfile=/path/to/my/pfile_add.ora`
`add_during_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to add during the upgrade. Example: `sales3.add_during_upgrade_pfile=/path/to/my/newpfile.ora`

Table 3-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`after_action`	(Optional) Specifies a custom action that you want to have performed after completing the upgrade job for the database identified by the prefix address.
	The script that you use must be in the form of *`name.ext`* (for example, `myscript.sh`, so that AutoUpgrade can identify the type of script that you want to run. Permitted extension options:
	• Unix shell (`.sh`)
	• Microsoft Windows batch (`.bat`, `.cmd`)
	• Microsoft Windows PowerShell (`.ps1`)
	• Oracle SQL file (`.sql`), with a local operation only designated by the prefix.
	By default, if the script fails, then AutoUpgrade continues to run. Use the `Y` flag to specify that AutoUpgrade stops if the operating system detects that your script fails. If the script finishes with a status different than `0`, then it is considered a failed completion.
	In contrast to the global `after_action` parameter, the local `after_action` parameter can specify a SQL script, which then runs on the database using the target Oracle Database binaries on a non-CDB Oracle home, or on `CDB$ROOT`. If you want to run additional container-specific actions, then they must be set within the code. For more complex scenarios, you can run container-specific actions in a shell.
	Examples:
	Run the specified script before AutoUpgrade starts processing, with the `Y` flag set to stop AutoUpgrade if the script fails:
	`sales2.after_action=/user/path/script.sh Y`
	Run the specified script before AutoUpgrade starts processing, with AutoUpgrade set to continue to run if the script fails:
	`sales3.after_action=/user/path/script.sh`

Table 3-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`before_action`	(Optional) Specifies a custom action that you want to have performed before starting the upgrade job for the specific database job addressed by the prefix. If you want to have a script run before all upgrade jobs, then specify that script by using the local parameter (`global.before_action`)
	
	The script that you use must be in the form of *`name.ext`* (for example, `myscript.sh`), so that AutoUpgrade can identify the type of script that you want to run. Permitted extension options:
	• Unix shell (`.sh`)
	• Microsoft Windows batch (`.bat`, `.cmd`)
	• Microsoft Windows PowerShell (`.ps1`)
	• Oracle SQL file (`.sql`), with a local operation only designated by the prefix.
	By default, if the script fails, then AutoUpgrade continues to run. Use the `Y` flag to specify that AutoUpgrade stops if the operating system detects that your script fails. If the script finishes with a status different than `0`, then it is considered a failed completion.
	
	In contrast to the global `before_action` parameter, the local `before_action` parameter can specify a SQL script, which can run on the database in the source database Oracle home, using the earlier release Oracle Database binaries. The script runs on a non-CDB Oracle home, or on `CDB$ROOT`. If you want to run additional container-specific actions, then they must be set within the code. For more complex scenarios, you can run container-specific actions in a shell.
	
	Examples:
	
	Run the specified script before AutoUpgrade starts processing, with the `Y` flag set to stop AutoUpgrade if the script fails:
	
	`sales.before_action=/user/path/script.sh Y`
	
	Run the specified script before AutoUpgrade starts processing, with AutoUpgrade set to continue to run if the script fails:
	
	`sales4.before_action=/user/path/script.sh`

Table 3-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`catctl_options`	(Optional) Specifies one or more of a set of `catctl.pl` options that you can select for AutoUpgrade to submit for `catctl.pl` to override default behavior. For a complete description of the options, refer to "Parallel Upgrade Utility (`catctl.pl`) Parameters," which is linked to at the end of this table. Available `catctl.pl` options: • `-n` Number of processes to use for parallel operations. • `-N` Number of SQL processors to use when upgrading PDBs. • `-t` Run SQL in classic upgrade overwriting default replay upgrade method • `-T` Takes offline user schema-based table spaces. • `-z` Turns on production debugging information for `catcon.pm`. Example: `sales4.catctl_options=-t`
`checklist`	(Optional) Specifies the path to a checklist that you can use to override the default list of fixups that AutoUpgrade performs, such as fixups that you do not want implemented automatically, due to policy or security concerns. To use this parameter during other AutoUpgrade modes, you must run AutoUpgrade in `analyze` mode. After AutoUpgrade finishes the analysis, you can then find the checklist file identified by the database name under the precheck directory (*dbname*`_checklist.cfg`). Update the file manually to exclude the fixups that you want AutoUpgrade to bypass, and save the file with a new name. When you run AutoUpgrade again, you can specify the parameter pointing to the checklist file that you created, and modify fixups that are performed for individual databases. If you do not specify a checklist file path, then the set of fixups that run during the upgrade is the latest version of the checklist file that is created during the deploy mode that you specified. Example: `sales.checklist=/u01/app/oracle/upgrade-jobs/salesdb_checklist.cfg` In the preceding example, `salesdb_checklist.cfg` is the checklist configuration file for the database `salesdb`.
`del_after_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to remove after the upgrade. Example: `sales3.del_after_upgrade_pfile=/path/to/my/pfile_del.ora`

Table 3-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
del_during_upgrade_pfile	(Optional) Specifies a path and file name of a PFILE whose parameters you want to have removed during upgrade. Example: sales3.del_during_upgrade_pfile=/path/to/my/oldpfile.ora
env	(Optional) Specifies custom operating system environment variables set on your operating system, excluding ORACLE_SID, ORACLE_HOME, ORACLE_BASE, and TNS_ADMIN. Use case: Use this parameter to provide environment setting that are indicated in the database sqlnet.ora file, such as secure socket layer cipher suites that are used for Oracle Wallet. Syntax: *prefix*.env=VARIABLE1=value1/, VARIABLE2=value2/ For example, assume that for the PDB sales2, the value for WALLET_LOCATION is set using custom environment variables: WALLET_LOCATION= (SOURCE= (METHOD=file) (METHOD_DATA=(DIRECTORY=/databases/ wallets/$CUSTOM_ENV1/$CUSTOM_ENV2)) In that case, for AutoUpgrade to know what those custom environment variables are, you must provide them using the env parameter, where *dir1* is the path indicated by the environment variable CUSTOM_ENV1, and *dir2* is the path specified by CUSTOM_ENV2: sales2.env=CUSTOM_ENV1=*dir1*,CUSTOM_ENV2=*dir2*

Table 3-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`log_dir`	(Optional with AutoUpgrade 19.8) Sets the location of log files that are generated for database upgrades that are in the set of databases included in the upgrade job identified by the prefix for the parameter.
	When set, AutoUpgrade creates a hierarchical directory based on a local log file path that you specify. For example, where the job identifier prefix is `sales`, and where `log_dir` is identified as `upgrade-jobs`, and *`stage1`*, *`stage2`*, and *`stagen`* represent stages of the upgrades:
	`/u01/app/oracle/upgrade-jobs`
	`/temp/`
	`/sales/`
	`/sales/`*`stage1`*
	`/sales/`*`stage2`*
	`/sales/`*`stagen`*
	You cannot use wild cards for paths, such as tilde (~). You must use a complete path.
	Example:
	`salesdb.log_dir=/u01/app/oracle/upgrade-jobs`
	By default, if the global configuration file parameter `global.autoupg_log_dir` is specified, and you do not specify `log_dir`, then the default is the path specified in `global.autoupg_log_dir`.
	When neither `global.autoupg_log_dir` nor `log_dir` is specified, then by default the log files are placed in the location indicated by the `orabase` utility for the databases that you include in your configuration file. In that case, the default logs directory is in the path *`ORACLE_BASE`*`/cfgtoollogs/autoupgrade`.
	If the `orabase` utility fails for all databases included in the configuration file, then the log file location is then based on the `temp` directory for the user running AutoUpgrade.

Table 3-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`pdbs`	(Optional) Sets a list of PDBs on which you want the upgrade to run. This parameter only applies to upgrades of multitenant architecture (CDB) databases. If you are plugging in and upgrading a non-CDB database, then this parameter is ignored. The PDB list is comma-deliminated. The list can contain either PDB names, or a star character (*), which indicates that you want to upgrade all PDBs that are open on the CDB at the time that you run AutoUpgrade. If a PDB is in a mounted state, then AutoUpgrade ignores that PDB when running in ANALYZE mode. If the parameter is not specified, then all PDBs in the CDB are upgraded. However, if the PDB is in mount state, and the deploy mode is fixups, deploy or upgrade, then the PDB is opened in read-write mode, or upgrade mode, or both, so that the stages can run. Example: `sales1.pdbs=pdb1, pdb2, pdb`*`n`* `  `*`upgr1`*`.pdbs=*`

`raise_compatible`	(Optional) Increases the compatible parameter to the default value of the target release after the upgrade is completed successfully.
	Options:
	`[yes | no]`
	The default value is `no`.
	CAUTION:
	• After the `COMPATIBLE` parameter is increased, database downgrade is not possible.
	• Oracle recommends that you only raise the `COMPATIBLE` parameter to the current release level after you have thoroughly tested the upgraded database.
	• Regardless of what value you use for the `autoupgrade` command-line parameter `restore`, if you set the value of the configuration file parameter `raise_compatible` to `yes`, then before starting the upgrade, you must delete manually any guaranteed restore point you have created. After the upgrade is completed successfully, AutoUpgrade deletes the guaranteed restore point it creates before starting the upgrade. When AutoUpgrade starts the POSTUPGRADE stage, there is no way to restore the database.
	Example:
	`sales1.raise_compatible=yes`

Table 3-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`remove_underscore_parameters`	(Optional) Removes underscore (hidden) parameters from `PFILE` files during upgrade, and after upgrade, for all Oracle Databases in the configuration file. Underscore parameters should only be used by advice of Oracle Support.
	Options:
	`[yes | no]`
	The default value is `no`.
	Example:
	`sales1.remove_underscore_parameters=yes`

`restoration`	(Optional) Generates a Guaranteed Restore Point (GRP) for database restoration. If you set `restoration=no`, then both the database backup and restoration must be performed manually. Use this option for databases that operate in `NOARCHIVELOG` mode, and for Standard Edition and SE2 databases, which do not support the Oracle Flashback technology feature Flashback Database.
	Options:
	`[yes | no]`
	The default value is `no`.
	Example:
	`sales1.restoration=no`

`run_utlrp`	(Optional) Enables or disables running `utlrp` as part of the upgrade. The `utlrp` utility recompiles all Data Dictionary objects that become invalid during a database upgrade. Oracle recommends that you run this utility after every Oracle Database upgrade. Options: `yes`, `no`. The default is enabled (`yes`). Example: `prefix.run_utlrp=yes`
`sid`	(Required) Identifies the Oracle system identifier (SID) of the database that you want to upgrade. Example: `sales1.sid=salesdb`
`skip_tde_key_import`	(Optional) The default is `NO`. You can use this option for nonCDB-to-PDB and unplug/plug operations. When set to `YES`, the import of the source database KeyStore import into the target database is skipped, without raising an error. AutoUpgrade will leave the PDB open in upgrade mode, so that you can import the keys manually yourself.

Table 3-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
source_home	(Required for analyze, fixups, and deploy modes. Optional for upgrade mode.) Current Oracle home (ORACLE_HOME) of the database that you want to upgrade. For the mode upgrade, the source home and target home values can be the same path. Example: sales2.source_home=*/path/to/my/source/oracle/home*
source_tns_admin_dir	(Optional) Specifies the path to the TNS_ADMIN directory in the source database home. This parameter has no effect on Microsoft Windows, because on Windows, the TNS_ADMIN environmental variable is set within the registry. Example: sales1.source_tns_admin_dir=/u01/app/oracle/12.2/dbhome01/network/admin

Table 3-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`start_time`	(Optional) Sets a future start time for the upgrade job to run. Use this parameter to schedule upgrade jobs to balance the load on your server, and to prevent multiple jobs from starting immediately.
	Values must either take the form `now` (start immediately), or take the English Date Format form *DD*/*MM*/*YYYY* or *MM*/*DD*/*YYYY*, where *MM* is month, *DD* is day, and *YYYY* is year. If you do not set a value, then the default is `now`.
	Example:
	`sales1.start_time=now`
	`sales2.start_time=07/11/2020 01:30:15`
	Permitted values:
	`now`
	`30/12/2019 15:30:00`
	`01/11/2020 01:30:15`
	`2/5/2020 3:30:50`
	If more than one job is started with the `start_time` value set to `now`, then AutoUpgrade schedules start times based on resources available in the system, which can result in start time for jobs that are separated by a few minutes.
	Values are invalid that use the wrong deliminator for the date or time element, or use the wrong date or hour format.
	Example:
	`30-12-2019 15:30:00`
	`01/11/2020 3:30:15pm`
	`2020/06/01 01:30:15`

`target_base`	(Optional) Specifies the target `ORACLE_BASE` path for the target Oracle home.
	Example:
	`target_base=/u01/app/oracle`
	`sales4.target_base=/u04/app/oracle4`

`target_cdb`	(Optional) Specifies the SID of the target CDB into which a non-CDB Oracle Database is plugged in. This parameter is mandatory when you want to upgrade and convert a non-CDB Oracle Database. Example: `emp.target_cdb=salescdb`

Table 3-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`target_pdb_copy_option=file_name_convert=('f1', 'r1', 'f2', 'r2', 'f3', 'r3'...)`	(Optional) This option is only used when creating a pluggable database within the target CDB. It specifies the `file_name_convert` option that will be used by the create pluggable database statement that is executed by AutoUpgrade when converting a non-CDB database to a PDB or an existing PDB from a different source CDB into a PDB in the specified target CDB. If you do not specify this parameter, then the default value of the parameter is `NOCOPY`, and existing data files are reused.
	On the target CDB, if you have the parameters `DB_CREATE_FILE_DEST` or `PDB_FILE_NAME_CONVERT` set, and you want these parameters on the target CDB to take effect, then set the value of *`prefix`*`.target_pdb_copy_option=file_name_convert=NONE`
	If you want one or more data file names changed during conversion, then enter values for the parameter to indicate the filename you want to change, and the filename to which you want the existing files copied, using the syntax *`prefix`*`.target_pdb_copy_option=('f1', 'r1', 'f2', 'r2', . . .)`, where *`f1`* is the first filename pattern on your source, *`r1`* is the first replacement filename pattern on your target CDB, *`f2`* is the second filename pattern on your source, *`r2`* is the second replacement file pattern on your target CDB, and so on.
	Example:
	In this example, AutoUpgrade will copy existing datafiles during conversion of a database specified with the prefix string `upg1` to replace the file path string and filename `/old/path/pdb_2` with the file path string and filename `/new/path/depsales`:
	`upg1.target_pdb_copy_option=file_name_convert=('/old/path/pdb_2', '/new/path/depsales')`
	To convert OMF files with `target_pdb_copy_option=file_name_convert`, the target Oracle home must be Oracle Database 19c Release Update 6 or later (19.6.0), or Oracle Database 18c Release Update 10 or later (18.10.0).
	In this example, the parameter is configured so that data files that are stored on Oracle ASM, but not stored as Oracle-managed files, are copied from `+DATA/dbname/sales` to `+DATA/dbname/depsales`:
	`upg1.target_pdb_copy_option=file_name_convert=('+DATA/dbname/sales', '+DATA/dbname/depsales')`

Table 3-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`target_pdb_name`	(Optional) Specifies the name that you want to assign to a non-CDB source Oracle Database after is plugged in to the target CDB. The default value is to use the database unique name of the non-CDB Oracle Database. If you want to specify a name that is different from the existing name of the non-CDB when you plug it in to the CDB, then you must set this parameter. Example: `emp.target_pdb_name=sales2`
`target_tns_admin_dir`	(Optional) Specifies the path to the TNS_ADMIN directory in the target database home. Example: `sales1.target_tns_admin_dir=/u01/app/oracle/19/dbhome01/network/admin`
`timezone_upg`	(Optional) Enables or disables running the time zone upgrade as part of the AutoUpgrade process. To preserve data integrity, Oracle recommends that you upgrade the time zone settings at the time of your database upgrade. In particular, upgrade the timezone when you have data that depend on the time zone, such as `timestamp with time zone` table columns. Note that this setting can be disabled by overwriting the fixup on the checklist file. Options: `yes`, `no`. The default is enabled (`yes`). Example: `sales1.timezone_upg=yes`

Table 3-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
upgrade_node	(Optional) Specifies the node on which the current user configuration is valid. The default value is localhost. The purpose of this parameter is to prevent AutoUpgrade from processing databases that are listed in the configuration file that you use with AutoUpgrade, where the value for the upgrade_node parameter does not correspond to the current host name. It does not enable running AutoUpgrade remotely. You can use the keyword localhost as a wild card to indicate that databases on the local host should be processed. Use case: The configuration file config.cfg contains 10 databases. Five of the databases have the value of upgrade_node set to denver01. The remaining five have the value of upgrade_node set to denver02. If AutoUpgrade is run on the server denver01 using the configuration file config.cfg, then AutoUpgrade only processes the databases where upgrade_node is set to denver01. It ignores the databases where upgrade_node is set to denver02. The utility hostname identifies the value used to resolve the upgrade node. Example: hostname denver02 sales1.upgrade_node=denver01

Locally Modifiable Global Parameters for AutoUpgrade Configuration File

Locally modifiable global parameters are parameters that you set both globally, and as you require, set locally, so that you can better control AutoUpgrade job processing.

Usage Notes

Locally modifiable global parameters are required parameters. You must define these parameters in your AutoUpgrade configuration file, either globally, or locally. With locally modifiable global parameters, you can use the prefix global to set values as global parameters for all jobs in your AutoUpgrade configuration file, but reset the same parameter with a local job prefix for a particular job in the same configuration file. You can also choose to set locally modifiable global parameters only as local parameters for each AutoUpgrade job.

> **Note:**
>
> These parameters are available in the latest version of AutoUpgrade that you can download from My Oracle Support.

When a locally modifiable global parameter is set both with a global prefix, and with a local job prefix, the locally modified parameter value overrides the global parameter values for the job identified by the prefix that you use with the parameter. The syntax you use is in the form `global.target_home=Global target Oracle home`, and `database.target_home=local target Oracle home`.

Example:

In the AutoUpgrade configuration file, the required parameter `target_home` is set globally to one Oracle home path. But in the configuration file, the same parameter is set locally to a different Oracle home path. As AutoUpgrade processes the jobs in the configuration file, it uses the locally defined path for `target_home` for the job defined by the prefix `upgrade3`, overriding the global parameter setting:

```
global.target_home=/u01/app/oracle/21.0.0/dbhome01
upgrade3.target_home=/u03/app/oracle3/12.2.0.1/dbhome3
```

Table 3-3 Locally Modifiable Global Parameters for AutoUpgrade Configuration Files

Parameter	Description
`defer_standby_log_shipping`	(Optional) Defers shipping logs from the primary database to the standby database before the upgrade, where you have a primary database with a physical standby database. When Autoupgrade defers log shipping, you will receive a notice that log shipping is deferred, and that after the upgrade completes successfully, you need to reenable shipping logs from the primary database to the secondary database. The default option is `Yes`. If you change the default to `No`, then log shipping will not be deferred.

`drop_grp_after_upgrade`	(Optional) Deletes the Guaranteed Restore Point (GRP) after database upgrade. If you select this option, then GRP is deleted after upgrade completes successfully.
	Options:
	`[yes | no]`
	The default value is `no`.
	Examples:
	`global.drop_grp_after_upgrade=yes`
	`sales.drop_grp_after_upgrade=yes`

Table 3-3 (Cont.) Locally Modifiable Global Parameters for AutoUpgrade Configuration Files

Parameter	Description
enable_local_undo	(Optional) For a CDB upgrade, specifies whether or not LOCAL undo should be enabled before the upgrade of CDB$ROOT by running the following statement: ALTER DATABASE LOCAL UNDO ON; The allowed values are [YES \| NO]. The default value is NO. When local undo is first enabled, the size of the undo tablespace in PDB$SEED is determined as a factor of the size of the undo tablespace in CDB$ROOT. The default is 30 percent of the undo tablespace size. Every other PDB in the CDB inherits this property from PDB$SEED. Ensure that there is enough space to allocate new UNDO tablespaces.
manage_network_files	Specifies whether network files are processed during the upgrade. Options: [FULL|SKIP|IGNORE_READ_ONLY] FULL: (default) Raise all exceptions encountered during the copy and merge of network files into the target Oracle home. SKIP: Do not process network files during postupgrade. IGNORE_READ_ONLY: Attempt to copy and merge network files, but do not raise an exception during the upgrade if the target file is read only The following network files are processed: oranfstab, ldap.ora, tnsnames.ora, sqlnet.ora, and listener.ora
remove_underscore_parameters	(Optional) Removes underscore (hidden) parameters from PFILE files during upgrade, and after upgrade, for all Oracle Databases in the configuration file. Underscore parameters should only be used by advice of Oracle Support. Options: [yes | no] The default value is no. Example: global.remove_underscore_parameters=yes

Table 3-3 (Cont.) Locally Modifiable Global Parameters for AutoUpgrade Configuration Files

Parameter	Description

`restoration`	(Optional, available with Enterprise Edition only) Generates a Guaranteed Restore Point (GRP) for database restoration. If you select this option, then both database backup and database restoration must be performed manually by the DBA.
	Options:
	`[yes | no]`
	The default value is `yes`.
	Example:
	`global.restoration=no`
	Standard Edition does not support Flashback Database, so this option is not available for Standard Edition. If your database is a Standard Edition Oracle Database, then you must ensure that you have a separate fallback mechanism is in place.

`target_version`	(Optional) Specifies the target release version on which you want AutoUpgrade to perform the upgrade. AutoUpgrade uses the release version information that you provide in this parameter to ensure that the correct checks and fixups are used for the target Oracle Database release to which you are upgrading. The format for this parameter are period-delimited values of valid Oracle versions.
	Valid values
	• 12.2
	• 18
	• 19
	• 21
	This option is only required if the target home is not present on the system, or if the target home is a 12.2 release. Otherwise, AutoUpgrade can derive the target release value.
	Example:
	`global.target_version=18`
	`employees.target_version=12.2`

Table 3-3 (Cont.) Locally Modifiable Global Parameters for AutoUpgrade Configuration Files

Parameter	Description

`target_home`	Specifies the target Oracle home (`ORACLE_HOME`) path.
	Example:
	`global.target_home=/u01/app/oracle/21.0.0/dbhome01`
	`sales4.target_home=/u04/app/oracle4/21.0.0/dbhome04`
	If the mode is `ANALYZE` or `FIXUPS`, then the parameter `target_home` is optional.

`target_base`	(Optional) Specifies the target `ORACLE_BASE` path for the target Oracle home.
	Example:
	`global.target_base=/u01/app/oracle`
	`sales4.target_base=/u04/app/oracle4`

AutoUpgrade and Oracle Database Configuration Options

When you run AutoUpgrade, it determines the type of database (Oracle Database, Oracle Database Standalone with Oracle ASM, or Oracle RAC), and performs an upgrade for that type of database

- Non-CDB to PDB Upgrade Guidelines and Examples
  Before conversion, back up your datafiles and database, and follow the guidelines for your source Oracle Database release.
- AutoUpgrade Process Flow for Oracle Grid Infrastructure Configurations
  When AutoUpgrade detects Oracle RAC, Oracle RAC One Node, or Oracle Restart, it proceeds to perform upgrade steps required for all Oracle RAC instances.
- Oracle RAC Requirements for Upgrade with AutoUpgrade
  To determine if AutoUpgrade can upgrade your Oracle Real Application Clusters (Oracle RAC) or Oracle RAC One Node database, review the use case requirements.
- AutoUpgrade and Oracle Data Guard
  The AutoUpgrade utility that you download from My Oracle Support can simplify the upgrade process for your primary and secondary databases configured for Oracle Data Guard.

Non-CDB to PDB Upgrade Guidelines and Examples

Before conversion, back up your datafiles and database, and follow the guidelines for your source Oracle Database release.

To ensure that no data is lost during the conversion, Oracle strongly recommends that allow time in your upgrade plan to implement your backup strategy before you use AutoUpgrade to perform a non-CDB upgrade and conversion.

Guidelines for Upgrade Planning

The non-CDB-to-PDB conversion and upgrade process is not recoverable. To ensure a proper upgrade and conversion, and to reduce unexpected downtime, Oracle strongly recommends that you address any error conditions found during the analyze phase.

If you do not set the `target_pdb_copy_option` in your AutoUpgrade configuration file, then the database conversion uses the same file location and file names that are used with existing database files. To prevent potential data loss, ensure that your data is backed up, and consider your file placement plans before starting AutoUpgrade.

GRP and Upgrades from Non-CDB to Multitenant Architecture

- During the upgrade, AutoUpgrade creates a guaranteed restore point (GRP) that is available only in the context of the upgrade stage of the AutoUpgrade Deploy workflow. To ensure against any potential data loss, you must implement your backup strategy before starting AutoUpgrade.
- Database conversion from non-CDB to the multitenant architecture is performed during the AutoUpgrade Drain stage. After this stage is complete, the GRP that AutoUpgrade creates is removed, and it is not possible to use the AutoUpgrade `restore` command to restore the database. In the event that you require a recovery to the earlier non-CDB Oracle Database release, you must be prepared to recover the database manually.

Example 3-2 Upgrading and Converting a Non-CDB to Oracle Database 19c Using Multitenant Architecture

During the Deploy conversion and upgrade workflow, AutoUpgrade version 19.9 and later creates a GRP, and runs the Prefixup stage. If any part of the Deploy workflow up to the Prefixup stage completion fails, then AutoUpgrade can restore the database back to the GRP created at the start of the deployment.

However, after the Prefixup stage is complete, the upgraded database is plugged in to the target release Oracle Database container database (CDB) to complete conversion. As soon as the non-CDB is plugged into the CDB, the GRP is no longer valid, and is dropped.

If anything goes wrong during the plug-in, then AutoUpgrade cannot recover and restore the database. You must restore the database manually.

AutoUpgrade Process Flow for Oracle Grid Infrastructure Configurations

When AutoUpgrade detects Oracle RAC, Oracle RAC One Node, or Oracle Restart, it proceeds to perform upgrade steps required for all Oracle RAC instances.

When you start AutoUpgrade, it detects when Oracle Database is configured with Oracle Grid Infrastructure, either as a cluster member node member in Oracle Real Application Clusters (Oracle RAC), or an Oracle RAC One Node configuration, or an Oracle Grid Infrastructure for a Standalone Server (Oracle Restart) configuration.

When AutoUpgrade detects Oracle Grid Infrastructure, it performs the following steps, in sequence:

1. It disables Oracle RAC, Oracle RAC One Node, or Oracle Restart services.
2. For Oracle RAC, it disables the cluster membership of all Oracle RAC database cluster member nodes in Oracle Clusterware.
3. It shuts down the database, or all instances of the Oracle RAC database.
4. For Oracle RAC, it starts the local Oracle Database instance with the cluster parameter set to `TRUE`. For Oracle RAC One Node or Oracle Grid Infrastructure for a standalone server, it starts the local database instance.
5. It updates the local instance to the new Oracle Database release binaries.
6. It starts `srvctl upgrade database` from the local Oracle Database instance home, and upgrades the configuration of the Oracle Grid Infrastructure services to the new release.
7. It enables Oracle Grid Infrastructure services for the database, using `srvctl enable database`. For Oracle RAC, it adds the upgraded Oracle RAC database to the Oracle RAC cluster as a cluster member node.
8. It recreates the server parameter file (`SPFILE`) with the updated parameters, and the parameter options you previously set for your Oracle Grid Infrastructure environment not affected by the release update.
9. It starts up the Oracle Database. For Oracle RAC It starts all instances of Oracle Real Application Clusters on the cluster.

> **Note:**
>
> Before you start AutoUpgrade on an Oracle Grid Infrastructure for a standalone server (Oracle Restart, Oracle RAC One Node, or Oracle RAC Database, you must upgrade Oracle Grid Infrastructure to a release equal to or more recent than the Oracle Database release to which you are upgrading.

Oracle RAC Requirements for Upgrade with AutoUpgrade

To determine if AutoUpgrade can upgrade your Oracle Real Application Clusters (Oracle RAC) or Oracle RAC One Node database, review the use case requirements.

Requirements for Using AutoUpgrade with Oracle RAC Databases

You can use AutoUpgrade to perform upgrades of Oracle RAC or Oracle Real Application Clusters One Node systems. However, your system must meet all of the following requirements:

- Must be either a Linux or Unix-based system. Microsoft Windows systems are not supported.
- Must meet the upgrade requirements to upgrade to the new Oracle Database release.
- Must be registered and managed through the Server Control (`SRVCTL`) utility.

Required Tasks for Database Administrators to Use AutoUpgrade

As the database administrator, you must complete the following tasks:

- Create an adequate backup strategy to prevent data loss from any problems resulting from the upgrade.
- Configure Listener and Transparent Network Substrate (TNS) files, both for local `tnsnames.ora` and SCAN listeners, if needed.
- Configure Oracle Wallet certificates and management (if needed), and configure for automatic login.

Related Topics

- Enabling Full Deployments for AutoUpgrade

AutoUpgrade and Oracle Data Guard

The AutoUpgrade utility that you download from My Oracle Support can simplify the upgrade process for your primary and secondary databases configured for Oracle Data Guard.

- How AutoUpgrade Performs Oracle Data Guard Upgrades
  AutoUpgrade can detect Oracle Data Guard configurations, and defer shipping logs to standby databases configured for the primary database.
- Steps AutoUpgrade Completes for Oracle Data Guard Upgrades
  The steps that AutoUpgrade completes vary, depending on whether standby databases are managed manually, or through Data Guard Broker.
- Steps After the Primary Database is Upgraded
  For Oracle Data Guard upgrades, after you upgrade the primary database you must complete these procedures.

How AutoUpgrade Performs Oracle Data Guard Upgrades

AutoUpgrade can detect Oracle Data Guard configurations, and defer shipping logs to standby databases configured for the primary database.

AutoUpgrade automatically detects the presence of an Oracle Data Guard deployment, and whether that deployment is configured manually, or uses Data Guard Broker to manage and monitor Oracle Data Guard configurations.

When you set the parameter `defer_standby_log_shipping` to `no` (the default) in the configuration file, AutoUpgrade can defer the log-shipping to configured standby databases, both when Oracle Data Guard is configured manually, and when Oracle Data Guard is configured through Data Guard Broker.

Preparation Before AutoUpgrade Upgrades of Databases with Oracle Data Guard

Before you begin the upgrade, to be prepared in case of a failure during the primary database upgrade, or in case the primary database must be reverted to the source Oracle home, ensure that your standby databases are protected and recoverable.

Steps AutoUpgrade Completes for Oracle Data Guard Upgrades

The steps that AutoUpgrade completes vary, depending on whether standby databases are managed manually, or through Data Guard Broker.

For Oracle Data Guard earlier release (**source**) databases where Oracle Data Guard is managed manually, or through Data Guard Broker, to manage log-shipping to standby databases, you can set `defer_standby_log_shipping=yes` in your AutoUpgrade configuration file (the default is `no`). However, the specific actions that AutoUpgrade takes vary, depending on how you manage standby databases.

> **Note:**
>
> For standby databases managed either manually or through Data Guard Broker, after the upgrade completes, you must run `ENABLE DATABASE` *`database-name;`* on each of the standby archive log destinations after successful upgrade on the primary database, and perform all steps needed to have standby databases upgraded through the redo log apply.

Manually Managed Oracle Data Guard Standby Databases

For Oracle Data Guard standby databases supported for direct upgrade, AutoUpgrade places in `DEFER` mode all `VALID` and `ENABLED` standby archive log destinations before starting the upgrade process for both manually managed and Data Guard Broker managed standby databases.

Data Guard Broker-Managed Oracle Data Guard Standby Databases

For Oracle Database releases supported for direct upgrade with Oracle Data Guard standby databases that are managed using Data Guard Broker, AutoUpgrade completes the following actions:

- The primary database state is set to `TRANSPORT-OFF` to all standby databases configured with Data Guard Broker
- The Data Guard Broker files are copied from the source Oracle home to the target Oracle home.

> **Note:**
>
> If the Data Guard Broker files are located outside of the Oracle home, then files are not found and copied.

Steps After the Primary Database is Upgraded

For Oracle Data Guard upgrades, after you upgrade the primary database you must complete these procedures.

- Ensure that redo transport is enabled on the primary database, so that the upgrade is applied to the standby databases.

- Check that the archives are applied, and that there is a minimal gap. Oracle recommends that Apply Lag and Transport Lag is not bigger than 5 minutes.

Example 3-3 Checking Redo Transport Service Status

To check the status of the redo transport services on the primary database, use the Data Guard broker command-line interface (`DGMGRL`) `LogXptStatus` monitorable property. For example:

```
DGMGRL> SHOW DATABASE 'sales1' 'LogXptStatus' ;
```

Example 3-4 Checking Apply Lag and Transport Lag

To check that the archives are applied, and verify that Apply Lag and Transport Lag is not bigger than 5 minutes, log in to the primary database and submit a SQL query similar to the following:

```
[oracle]$ sqlplus / as sysdba
SYS@sales1>

SET LINESIZE 200
COL VALUE FOR A30
SELECT NAME,VALUE,TIME_COMPUTED,DATUM_TIME FROM V$DATAGUARD_STATS WHERE
NAME LIKE '%lag';
```

The result should be similar to this output:

```
NAME VALUE TIME_COMPUTED DATUM_TIME
------------- -------------- --------------------- --------------------
transport lag +00 00:00:00 timestamp timestamp
apply lag +00 00:01:07 timestamp timestamp
```

Related Topics

- Scenario 13: Monitoring a Data Guard Configuration

AutoUpgrade Configuration File Examples

Use these examples to understand how you can modify your own AutoUpgrade configuration files to perform a variety of configuration actions during the upgrade.

- Updating the TDE Wallet Store Location During Upgrade Using AutoUpgrade
  See how you can use AutoUpgrade configuration file parameters to update your Transparent Data Encryption (TDE) wallet store during upgrade.
- AutoUpgrade Configuration File with Two Database Entries
  See how you can specify upgrade options for multiple databases in a configuration file.
- Standardizing Upgrades With AutoUpgrade Configuration File Entries
  See how to enforce standardization of your database configurations during upgrades using AutoUpgrade.
- AutoUpgrade Configuration File for Incremental Upgrade of a Set of PDBs
  See how you can selectively upgrade a subset of PDBs using AutoUpgrade, without affecting the other PDBs on the source CDB.

- How to Run AutoUpgrade in a Script or Batch job
  Learn how to run AutoUpgrade in your own scripts in noninteractive mode by calling AutoUpgrade using the `noconsole` parameter.

Updating the TDE Wallet Store Location During Upgrade Using AutoUpgrade

See how you can use AutoUpgrade configuration file parameters to update your Transparent Data Encryption (TDE) wallet store during upgrade.

In previous releases, if you used Oracle Wallet with TDE, then you specified the location of the existing keystore directory location by using the deprecated `sqlnet.ora` parameter `SQLNET.ENCRYPTION_WALLET_LOCATION`. In Oracle Database 19c and later releases, you should specify the keystore location by using the `WALLET_ROOT` system parameter in the database initialization parameter file (`PFILE`). What you need to do depends on how your source Oracle Database release is configured:

- If your source Oracle Database release has `WALLET_ROOT` set already, then the parameter files that AutoUpgrade generates automatically pick up the `WALLET_ROOT` system parameter from the source database during the upgrade, and use that parameter in target database parameter files.
- If your source Oracle Database release does not have the initialization parameter `WALLET_ROOT` set, then you can use AutoUpgrade to complete that task during the upgrade.

1. Create a text file on your operating system with the `WALLET_ROOT` initialization parameter value for the directory that you want to use, and that provides the configuration option you want for the `TDE_CONFIGURATION` dynamic initialization parameter to create the type of keystores that you require. For example, if you configure `TDE_CONFIGURATION` to use `FILE` for Transparent Data Encryption software keystores, then Oracle Database creates the software keystore in `WALLET_ROOT/tde` (lower case).
2. In the AutoUpgrade configuration file, use the AutoUpgrade configuration file parameters `add_during_upgrade_pfile` and `add_after_upgrade_pfile` to refer to that file on the operating system to set `WALLET_ROOT` and `TDE_CONFIGURATION` during the upgrade.

For example, if you want `WALLET_ROOT` to use the path `$ORACLE_BASE/admin/hr/wallet`, and Transparent Data Encryption to store software keystores in the location `WALLET_ROOT/tde`, then you can create a text file called `tde-upgrade`, which contains the following lines:

```
WALLET_ROOT=$ORACLE_BASE/admin/hr/wallet
tde_configuration="KEYSTORE_CONFIGURATION=FILE"
```

You can then specify for AutoUpgrade to set these parameters in the AutoUpgrade configuration file. For example, to set the Transparent Data Encryption keystore during and after the upgrade, as part of the AutoUpgrade operation, add the following line to your local configuration file to call that text file:

```
#
#      Example local pfile configuration entries
```

```
upg1.add_after_upgrade_pfile=/usr/home/oracle/tde-upgrade
upg1.add_during_upgrade_pfile=/usr/home/oracle/tde-upgrade
```

Related Topics

- How Configuring Transparent Data Encryption Works

AutoUpgrade Configuration File with Two Database Entries

See how you can specify upgrade options for multiple databases in a configuration file.

This example is of an AutoUpgrade configuration file that specifies the upgrade of two databases. The configuration file specifies that AutoUpgrade performs the following actions:

Database 1

- An in-place database upgrade of the Oracle Database 12c Release 2 (12.2) CDB, where the source and target Oracle homes use the same Oracle Base directory (the database home directory for Oracle Database installation owner `oracle` (`/u01/app/oracle/`) on the same server hardware, with the same system identifier (sid=HR1).
- During the upgrade, all the PDBs of the CDB are upgraded (`pdbs=*`)
- The upgrade starts immediately (`start_time=now`)
- The database upgrade logs will be sent to the path `/database/logs/hr` (`log_dir=/database/logs/hr`)
- The Time Zone upgrade will run on all the containers (`timezone_upg=yes`)

Database 2

- An in-place database upgrade of the Oracle Database 18c CDB, where the source and target Oracle homes use the same Oracle Base directory (the database home directory for Oracle Database installation owner `oracle` (`/u01/app/oracle/`) on the same server hardware, with the same system identifier (`sid=SALES1`).
- The upgrade starts immediately (`start_time=now`)
- The database upgrade logs will be sent to the path `/database/logs/sales` (`log_dir=/database/logs/sales`).
- The Time Zone upgrade will not run on any containers (`timezone_upg=no`).

For both databases:

- The parameter `upgrade_node` specifies the actual system host name (`nodename-1`), and not to an alias assigned to the host name. (You can also use the keyword `localhost` to refer to the current system.)
- The global AutoUpgrade log files (also known as job manager logs) are placed under the path `/database/jobmgr` (`autoupg_log_dir=/database/jobmgr`).

```
#
# Global logging directory pertains to all jobs
#
global.autoupg_log_dir=/database/jobmgr

#
```

```
# Database 1
#
upg1.source_home=/u01/app/oracle/product/12.2.0.2/dbhome_1
upg1.target_home=/u01/app/oracle/product/21.0.0/dbhome_1
upg1.sid=HR1
upg1.start_time=now
upg1.pdbs=*
upg1.log_dir=/database/logs/hr
upg1.upgrade_node=nodename1
upg1.run_utlrp=yes
upg1.timezone_upg=yes
upg1.target_version=21

#
# Database 2
#
upg2.source_home=/u01/app/oracle/product/18.0.0/dbhome_1
upg2.target_home=/u01/app/oracle/product/21.0.0/dbhome_1
upg2.sid=SALES1
upg2.start_time=now
upg2.log_dir=/database/logs/sales
upg2.upgrade_node=nodename1
upg2.timezone_upg=no
upg2.target_version=21
```

Standardizing Upgrades With AutoUpgrade Configuration File Entries

See how to enforce standardization of your database configurations during upgrades using AutoUpgrade.

In the following configuration file, you can see how you can use AutoUpgrade configuration file entries to standardize their database configurations. The global `PFILE` entries are applied to all databases within the configuration file. The local `PFILE` entries are applied only to a specific database in the configuration file. The syntax for these `PFILE` values follow the same Oracle rules for `PFILE` configurations.

```
#
#    Example global pfile configuration entries
#
global.del_during_upgrade_pfile=/database/pfiles/
global_during_delinit.ora
global.add_during_upgrade_pfile=/database/pfiles/
global_during_addinit.ora
global.del_after_upgrade_pfile=/database/pfiles/global_after_delinit.ora
global.add_after_upgrade_pfile=/database/pfiles/global_after_addinit.ora

#
#      Example local pfile configuration entries
#
upg1.del_during_upgrade_pfile=/database/pfiles/hr_during_delinit.ora
upg1.add_during_upgrade_pfile=/database/pfiles/hr_during_addinit.ora
upg1.del_after_upgrade_pfile=/database/pfiles/hr_after_delinit.ora
upg1.add_after_upgrade_pfile=/database/pfiles/hr_after_addinit.ora
```

During the AutoUpgrade process, the files `during_upgrade_pfile_dbname.ora` and `after_upgrade_pfile_dbname.ora` are both created. These files are used to start the database during the upgrade, and after the upgrade. If you want to change a system parameter during the upgrade, or after the upgrade, then you can modify both files.

The global `PFILE` entries are applied first, and then the local `PFILE` entries designated by the job prefix `upgl` are applied. Within those two configuration files, entries in the parameter `del_upgrade_pfile` are applied first, followed by entries in the parameter `add_upgrade_pfile`. The parameters in these `PFILE` configuration entries are applied directly either to the `PFILE during_upgrade_pfile_dbname.ora` or to the `PFILE after_upgrade_pfile_dbname.ora`, depending on which `PFILE` is targeted.

Actions:

- `del_during_upgrade_pfile` Removes entries from `during_upgrade_pfile_dbname.ora`
- `add_during_upgrade_pfile` Add entries to `during_upgrade_pfile_dbname.ora`.
- `del_after_upgrade_pfile` Removes entries from `after_upgrade_pfile_dbname.ora`
- `add_after_upgrade_pfile` Add entries to `after_upgrade_pfile_dbname.ora`.

The files referenced by the parameters `del_during_upgrade_pfile` and `del_after_upgrade_pfile` have a single database parameter listed on each line. You cannot add any prefix to the parameter, because the entire line is part of the parameter name. Consider the following example:

```
#
# global.del_during_upgrade_pfile
#
processes
*.open_cursors
```

The result of this configuration setting is to remove from the `PFILE` for each database listed in the configuration file all references to the `processes` parameter, but not references to the `open_cursors` parameter: Only instances of `open_cursors` that have a prefix are removed. However, the parameters removed from the `PFILE` includes all parameters that are prefixed. For example, `*.processes` and `instance_name.processes` are both removed with this syntax.

The files referenced by the parameters `add_during_upgrade_pfile` and `add_after_upgrade_pfile` have a single parameter listed on each line with the format *`parameter=value`*. If you delete the entry from the `PFILE`, then the value field can be left empty. If the parameter is prefixed with `*.` or `instancename.`, then those references are not added to the modified `PFILE`. To update the value of an existing parameter, you must first delete it. You can then add the parameter with the desired value. Consider the following example:

```
#
# global.add_during_upgrade_pfile
#
processes=400
*.open_cursors=250
```

This global configuration file entry results in adding the following entries to the PFILE for each database that is listed in the configuration file:

```
processes=400
open_cursors=250
```

The parameter after_upgrade_pfile_dbname is used to create the database SPFILE during the postupgrade process.

AutoUpgrade Configuration File for Incremental Upgrade of a Set of PDBs

See how you can selectively upgrade a subset of PDBs using AutoUpgrade, without affecting the other PDBs on the source CDB.

In this scenario, you upgrade two specific PDBs, without upgrading the other PDBs in the source CDB, To perform the incremental upgrade, you direct AutoUpgrade in the configuration file to unplug the PDBs you specify from an earlier release CDB, plug them into a target release CDB, and then upgrade the earlier release PDBs on the target CDB. This selection of PDBs to unplug, plug in, and upgrade, enables you to perform an incremental upgrade of PDBs on the earlier release CDB to reduce downtime.

The following configuration file identifies the CDB CDB122 as the source CDB. The source CDB has 10 PDBs, PDB1 through PDB10, but only PDB1 and PDB2 are upgraded. During the upgrade, the PDB named PDB2 has its name changed to DEPSALES, and the database file names for PDB2 are changed to DEPSALES:

```
global.autoupg_log_dir=/home/oracle/autoupg
upg1.sid=CDB122
upg1.source_home=/u03/app/oracle/product/12.2.0/dbhome_1
upg1.target_home=/u01/app/oracle/product/21.0.0/dbhome_1
upg1.target_cdb=CDB21C
upg1.pdbs=PDB2, PDB1
upg1.target_pdb_name.PDB2=DEPSALES
upg1.target_pdb_name.PDB1=EMPLOYEES
upg1.target_pdb_copy_option.PDB2=file_name_convert=('PDB2','DEPSALES')
```

This configuration file directs AutoUpgrade to do the following:

- Select PDBs from the source Oracle Database CDB122 in the home /u03/app/oracle/product/12.2.0/dbhome_1
- Upgrade PDBs PDB2 and PDB1 to the target Oracle Database 21c Oracle home /u01/app/oracle/product/21.0.0/dbhome_1
- Change the name of PDB2 to DEPSALES, and copy the PDB2 files using the new filename DEPSALES.
- Change the name of PDB1 to EMPLOYEES.

How to Run AutoUpgrade in a Script or Batch job

Learn how to run AutoUpgrade in your own scripts in noninteractive mode by calling AutoUpgrade using the `noconsole` parameter.

By default, AutoUpgrade runs in console mode, which enables you to run commands to monitor specific aspects of your AutoUpgrade jobs while they are running on your systems.

> **Note:**
>
> You can run only one AutoUpgrade instance at a time that is associated with a given configuration file.

Example 3-5

In this example, AutoUpgrade is run in Deploy mode, using the settings specified in the configuration file `autoupgrade.cfg`, and turning off console using the `noconsole` parameter.

```
java -jar autoupgrade.jar -config autoupgrade.cfg -mode deploy -
noconsole
```

Using the `noconsole` mode turns off requirements for user input, so you can enter this command in a script to run the upgrades you specify in the configuration file.

AutoUpgrade Internal Settings Configuration File

Internal configuration settings control how AutoUpgrade runs.

Usage Notes

These configuration settings are provided for reference only. Typically, you should not use these parameters.

Table 3-4 Internal Settings Configuration File Parameters for AutoUpgrade

Parameter	Default	Description
`heartbeatHeartbeatSleep`	1	Number of minutes to wait between each job heartbeat.
`heartbeatHeartbeatRetries`	10	Number of times to retry after a failed job heartbeat.
`shutdownJobWaitTime`	10	Number of minutes to wait before a running job is terminated in the job queue during a scheduled upgrade.
`systemChecksAbort_timer`	60	Number of minutes to wait before the system checks job is automatically terminated.

Table 3-4 (Cont.) Internal Settings Configuration File Parameters for AutoUpgrade

Parameter	Default	Description
systemChecksOracleHomeReqSpac	6g	Minimum adequate disk space (in GB) system check. (g is required.).
systemChecksMinCpuIdlePct	10	Warning alert threshold percentage to indicate that the remaining available percentage of CPU resources on the system can be inadequate to complete the upgrade.
systemChecksMinFreeMemPct	5	Warning alert threshold percentage to indicate that the remaining available percentage of system random access memory (RAM) can be inadequate to complete the upgrade.
systemChecksMinFreeSwapPct	5	Warning alert threshold percentage to indicate that the remaining available percentage of system swap space memory can be inadequate to complete the upgrade.
dbPreCheckAbortTimer	60	Number of minutes to wait before the database preupgrade checks job is automatically terminated.
dbUpgradeDurationTimer	180	Number of minutes to wait before the database upgrade job starts additional monitoring of the upgrade progress.
dbUpgradeWakeupTimer	3	Number of minutes to wait before the database upgrade job restarts monitoring the upgrade.
dbUpgradeAbortTimer	1440	Number of minutes to wait before the database upgrade is automatically terminated.
dbUpgradeFatalErrors	ORA-00600, ORA-07445	Identifies which upgrade internal errors automatically cause a post-upgrade restore of the database back to the guarantee restore point. Entries are comma-delimited.
dbPostUpgradeAbortTimer	60	Number of minutes to wait before the postupgrade job is automatically terminated.
dbRestoreAbortTimer	120	Number of minutes to wait before the Restoration job is automatically terminated.
dbGrpAbortTimer	3	Number of minutes to wait before the guarantee restore point job is automatically terminated.

AutoUpgrade Log File Structure

The AutoUpgrade utility produces a log file structure that includes job status and configuration files.

AutoUpgrade Log File Base Path

The AutoUpgrade log file path is set using the global parameter `autoupg_log_dir`. By default, the global parameter has the following definition:

```
global.autoupgr_log_dir=/database/jobmgr
```

AutoUpgrade configuration and status file paths are relative to the directory path that you establish with `global.autoupgr_log_dir`.

/cfgtoollogs/upgrade/auto

The automatic configuration tools log directory (`/cfgtoollogs/upgrade/auto`) contains three trace log files that provide specific information about each job that the AutoUpgrade job manager processes:

- `autoupgrade.log`: Provides detailed logs of the job that identify any problems that occur during job runs.
- `autoupgrade_usr.log`: Job information, which is formatted to enhance readability.
- `autoupgrade_err.log`: A report of any unexpected exceptions that occur when the job runs.

If problems occur when jobs start or stop, then you can use information in these log files to determine the cause of problems.

/config_files

The `config_files` directory contains AutoUpgrade internal runtime configurations and global temporary files.

/status

The `/status` directory contains JSON job status files. It contains two directories:

- `status.json`: This directory contains the final job status of all jobs completed in the JSON file format.
- `progress.json`: This directory contains the progress of all jobs currently running in the JSON file format.

Each module in the directories contains a status file for the operation that it performed. The module takes the following format, where the prefix `dbname` is the database name, *`operation`* is the upgrade operation that was performed, and the suffix *`status`* is the completion status of that operation:

```
dbname_operation-name.status
```

The success or the failure of that operation is indicated by the suffix, which is either `.success`, indicating the successful performance of that operation, or `.failure`,

indicating the failure of that operation. For example, the following module name indicates a successful run of the prechecks operation on the database `sales`:

```
sales_prechecks.success
```

The operation module name can be one of the following:

- `preupgrade`: The preupgrade stage, in which custom scripts can be run.
- `prechecks`: The upgrade checks completed before starting the upgrade.
- `grp`: Guaranteed restore point (using Oracle Flashback technology).
- `prefixups`: The preupgrade fixups run before starting the upgrade.
- `drain`: The stage where existing jobs are completed or migrated before starting the upgrade.
- `dbupgrade`: The stage in which the upgrade takes place.
- `postchecks`: The stage in which postupgrade checks are run after the upgrade is completed.
- `postfixups`: The stage in which postupgrade fixups are run.
- `postupgrade`: The stage in which custom postupgrade scripts can be run.

Individual Job and Database Log File Directories

Each job started by the AutoUpgrade dispatcher is given a directory with that job identifier prefix. Inside that job directory, each database in the job is given a log directory in the path `/database/logs/`*`sid`*, where *`sid`* is the system identifier for the database. For example, where the job identified in the configuration file is `sales1`, and the database system identifier is `sales`, the log file for the database sales is in the following path:

```
sales.log_dir=/database/logs/sales1
```

The log directory contain all the relevant log files for all the tasks performed for that database. By default, a directory identified by SID is created under the `/database/logs` directory. Each database job can have a separate log directory, if you choose to set up your configuration file that way.

`/####` (Job Number)

Individual job runs are placed in subdirectories identified by the run number, in the format `/####`, where `####` represents the job run number. For example: `0004`. Job run number directories contain the following log files:

- `autoupgrade_err.log`: Reports any unexpected exceptions that occur while the job runs.
- `autoupgrade`*`YYYYMMDD`*`.log`: AutoUpgrade trace log file. Provides detailed logs of the job that identify any problems that occur during job runs. The variable *`YYYYMMDD`* represents year, month, and day of the job.
- `autoupgrade_`*`YYYYMMDD`*`_user.log`: AutoUpgrade job status file, which is formatted to enhance readability. The variable *`YYYYYYMMDD`* represents year, month, and day of the job.

`/preupgrade`

The preupgrade directory (`/preupgrade`) contains the following files and log files:

- `prechecks_`*`databasename`*`.log`: Trace log file. This file provides detailed logs that can assist with identifying problems that occur during the preupgrade job stage. The variable *`databasename`* is the name of the database checked.
- *`databasename`*`_preupgrade.html`: HTML report on the database status. The variable *`databasename`* is the name of the database checked.
- *`databasename`*`_preupgrade.log`: Text report on the database status. The variable *`databasename`* is the name of the database checked.

`/dbupgrade`

The database upgrade directory (`/dbupgrade`) contains all log files associated with the database upgrade:

- `autoupgrade`*`YYYYMMDDHHMISCdbname`*`.log`: Log files for the source database, identified by the date on which the upgrade was run, and by the database name, indicating parallelism. Format:
  - *`YYYY`*: Year
  - *`MM`*: Month
  - *`DD`*: Day
  - *`HH`*: Hour
  - *`MI`*: Minute
  - *`SC`*: Second
  - *`dbname`*: Database name, where *`dbname`* is the database name.
- `catupgrade`*`YYYYMMDDHHMISCdbnameN`*`.log`: log files for the source database, identified by the date on which the upgrade was run.

  Format:
  - *`YYYY`*: Year
  - *`MM`*: Month
  - *`DD`*: Day
  - *`HH`*: Hour
  - *`MI`*: Minute
  - *`SC`*: Second
  - *`dbnameN`*: Database name, where *`dbname`* is the database name, and *`N`* indicates the parallelism: 0...3 for CDB ROOT, and Non-CDB databases, and 0...1 for PDBs.

`/temp`

Temporary AutoUpgrade files (`/temp`). This directory can contain files such as the `PFILE` used during an upgrade.

Enabling Full Deployments for AutoUpgrade

To enable a guaranteed restore point (GRP) so that you can flashback an upgrade, you must set up archive logging, and you should complete other tasks to enable AutoUpgrade to complete the upgrade.

For AutoUpgrade to be able to perform a full deployment of the new release Oracle Database, the following must be true:

- The database must have a proper configuration of the fast recovery area (FRA). Specifically, `DB_RECOVERY_FILE_DEST` and `DB_RECOVERY_FILE_DEST_SIZE` must be set, and be properly sized.
- Your source Oracle Database must be running in `ARCHIVELOG` mode.

> **Note:**
>
> AutoUpgrade creates a guaranteed restore point (GRP) during Deploy processing mode. You do not need to have a previously defined guaranteed restore point.

Example 3-6 Setting up Archive Logging and Fast Recovery Area (FRA) Before Using AutoUpgrade

In the following example, *`your-directory-or-diskgroup`* is the directory path or disk group where your recovery area is placed. The value for `DB_RECOVERY_FILE_DEST_SIZE` is specified as 50GB, but you should use the value that you require for your recovery area.

```
sqlplus / as sysdba
shutdown immediate;
startup mount;
alter system set db_recovery_file_dest_size = 50g scope=both sid='*';
alter system set db_recovery_file_dest ='your-directory-or-diskgroup'
scope=both sid='*';
alter database archivelog;
alter database open;
alter pluggable database all open;
```

Example 3-7 Password Files and Security Password File Updates

During the upgrade, the AutoUpgrade utility copies the password file from the source Oracle Database Oracle home to the target Oracle Database Oracle home. However, the copied password file retains the earlier release password file version. Oracle recommends that you regenerate the password file to update it to the new release password file version.

Example 3-8 Transparent Data Encryption and AutoUpgrade

To enable the AutoUpgrade utility to obtain the privileges required for copying transparent data encryption keystores, you must enable auto-login for these keystores

so that AutoUpgrade can copy them to the target release Oracle home. If you do not enable auto-login, then AutoUpgrade cannot complete the upgrade.

In addition, AutoUpgrade looks for the location of the Oracle Net administration directory using the `TNS_ADMIN` environment variable. If your `TNS_ADMIN` environment variable is not defined, then the path to the network administration directory defaults to the path *`Oracle_home`*`/network/admin`. If you need to specify a different path, then specify the path in your configuration file using the local configuration parameter `source_tns_admin_dir`, and if necessary, the target path with `target_tns_admin_dir`.

Enable an Auto-Login or a Local Auto-Login Software Keystore by using the ADMINISTRATOR KEY MANAGEMENT or SYSKM privilege on your existing keystore. For example, to create an auto-login software keystore of the password-protected keystore that is located in the `/etc/ORACLE/WALLETS/orcl` directory:

```
ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE
FROM KEYSTORE '/etc/ORACLE/WALLETS/tde'
IDENTIFIED BY password;

keystore altered.
```

> **Note:**
>
> After AutoUpgrade completes copying the transparent data encryption keystores, disable auto login, so that your previous release security is restored.

Related Topics

- Creating an Auto-Login or a Local Auto-Login Software Keystore
- Overview of Local Naming Parameters
- *Oracle Database Enterprise User Security Administrator's Guide*
- DB_RECOVERY_FILE_DEST

Examples of How to Use the AutoUpgrade Console

The AutoUpgrade console provides a set of commands to monitor the progress of AutoUpgrade jobs. The console starts by default when you run the AutoUpgrade utility, and is enabled or disabled by the parameters `console` and `noconsole`.

In console mode, the AutoUpgrade console enables you to run commands to monitor specific aspects of your AutoUpgrade jobs while they are running on your systems.

> **Note:**
>
> If the AutoUpgrade console is exited out before it completes, then the jobs that are running stop, and the job that are scheduled do not start. For this reason, do not exit the console or stop the AutoUpgrade process until all of the AutoUpgrade jobs are completed.

Example 3-9 Example of How to Enable and Disable the AutoUpgrade Console

> **Note:**
>
> You can run only one AutoUpgrade instance at a time that is associated with a given configuration file.

In this example, AutoUpgrade is run in Analyze mode, using the configuration file in `noconsole` mode.

```
java -jar autoupgrade.jar -config autoupgrade.cfg -mode analyze -
noconsole
```

Using the `noconsole` mode turns off requirements for user input, so it is suitable for use with scripts.

In this example, AutoUpgrade is run in Analyze mode, and the console is turned on again with the `-console` option:

```
java -jar autoupgrade.jar -config autoupgrade.cfg -mode analyze -console
```

Console user input is again resumed.

How to Override Default Fixups

You can use the `RUNFIX` column entry to disable automated fixups, except in cases where disabling the fixup violates security or Oracle policy.

The default fixups that are part of the AutoUpgrade procedure are generated during the Analyze processing mode stage. You can modify the generated fixups list to disable automatic fixups, so that you can run your own fixups.

The sequence of steps is as follows:

1. Run the AutoUpgrade utility in Analyze mode
2. Open and edit the *sid*`_checklist.cfg` file that is generated during Analyze mode, so that the fixups you want to do manually are disabled from running automatically.
3. In your most recent AutoUpgrade configuration file, under your local parameters list for the job on which you want to suppress the automatic fixup, find the parameter *sid*`.checklist`, where *sid* is the system identifier (SID) of the database on which you want to suppress an automatic fixup. By default, AutoUpgrade uses

the most recent generated file. If you want to point it to a different configuration file, then edit the parameter to provide a path to the `checklist.cfg` file that you have edited.

> **Note:**
>
> AutoUpgrade resume always uses the most recent *sid*`_checklist.config` file.
>
> For example, if you have two generated AutoUpgrade configuration files, `/logdir/100/`*sid*`_checklist.cfg`, and `/logdir/101/sid_checklist.cfg`, then you must either specify a direct path to the configuration file that you want to use, or edit the most recent file, which in this case is the *sid*`_checklist.cfg` file in `/logdir/101/`
>
> Suppose you have corrected an issue manually that you found, and want to have AutoUpgrade to use a fixup file with different checks. If you want to direct AutoUpgrade to use a different file, then you can specify the file path directly in the `checklist.cfg` file by using the *prefix*`.checklist` parameter, where *prefix* is the identifier for the database. For example: `prefix.checklist=logdir/repress-standard/sid_checklist.cfg`
>
> If the AutoUpgrade utility finds an error level database condition, and there is not a fixup available for it, or you have manually disabled the available fixup, then the AutoUpgrade job that contains the database with the error condition stops.

Use the examples that follow to assist you with this procedure.

Example 3-10 Starting Up the AutoUpgrade Utility in Analyze Mode

```
java -jar autoupgrade.jar -config config.cfg –mode analyze
```

Example 3-11 Creating a New Checklist for a Configuration File

In this scenario, you are running AutoUpgrade checks on an Oracle Database 11g Release 2 (11.2) database home, in preparation for an upgrade to Oracle Database 18c, with release update 8. You start with the following configuration file, called `config.cfg`:

```
global.autoupg_log_dir=/home/oracle/autoupg
upg1.sid=db11204
upg1.source_home=/databases/ee/product/11.2.0/dbhome_1
upg1.target_home=/databases/ee/product/18x/dbhome_1
upg1.target_base=/databases
upg1.target_version=18.8.0
```

You then complete the following steps:

1. Run the command `java -jar autoupgrade.jar -config config.cfg -mode analyze`

The command produces a checklist file in the following path:

```
/home/oracle/autoupg/db11204/100/prechecks/db11204_checklist.cfg
```

2. Move the checklist file to another location. For example:

```
oracle@example: $ cd $PRECHECKS
oracle@example: $ pwd
/home/oracle/autoupg/db11204/100/prechecks
oracle@example: $ mv ./db11204_checklist.cfg /tmp
```

3. Use a text editor to open up the file, and look for the checks `AMD_EXISTS` and `EM_PRESENT`.

   For example:

```
[SID] [db11204]
========================================
[container] [db11204]
========================================
[checkname] AMD_EXISTS
[stage] PRECHECKS
[fixup_available] YES
[runfix] YES
[severity] WARNING
--------------------------------------------------
[checkname] DICTIONARY_STATS
[stage] PRECHECKS
[fixup_available] YES
[runfix] YES
[severity] RECOMMEND
--------------------------------------------------
[checkname] EM_PRESENT
[stage] PRECHECKS
[fixup_available] YES
[runfix] YES
[severity] WARNING
--------------------------------------------------
...
[truncated]
```

4. Change the values for checks `AMD_EXISTS` and `EM_PRESENT` from `yes` to `no`. For example

```
[SID] [db11204]
========================================
[container] [db11204]
========================================
[checkname] AMD_EXISTS
[stage] PRECHECKS
[fixup_available] YES
[runfix] NO
[severity] WARNING
```

```
-------------------------------------------------
[checkname] DICTIONARY_STATS
[stage] PRECHECKS
[fixup_available] YES
[runfix] YES
[severity] RECOMMEND
-------------------------------------------------
[checkname] EM_PRESENT
[stage] PRECHECKS
[fixup_available] YES
[runfix] NO
[severity] WARNING
-------------------------------------------------
...
[truncated]
```

Notice that with both parameters, the `fixup_available` value is `YES`. That means that there is a fixup available, which you choose not to run. If no fixup is available, then the value for `runfix` is `N/A`.

5. Change the location where AutoUpgrade looks for the configuration file by updating the path for the parameter checklist. To do this, add an entry to the configuration file with the checklist pointer to the directory where your edited file resides. For example:

```
global.autoupg_log_dir=/home/oracle/autoupg
upg1.sid=db11204
upg1.source_home=/databases/ee/product/11.2.0/dbhome_1
upg1.target_home=/databases/ee/product/18x/dbhome_1
upg1.target_base=/databases
upg1.target_version=18.8.0
upg1.checklist=/home/oracle/db11204_checklist.cfg
```

6. Run the fixups using the configuration file that you have edited and moved. For example:

```
java -jar autoupgrade.jar -config config.cfg -mode fixups
```

Autoupgrade uses the configuration file in /home/oracle to run the AutoUpgrade utility.

Example 3-12 Find and Edit `checklist.cfg`

The Analyze mode generates a fixup file with the file name `checklist.cfg`. Navigate to the file, where *`DATABASE_LOGS_DIR`* is the value set for the AutoUpgrade `log_dir` parameter of the database, *`job-id`* refers to the job identifier that the AutoUpgrade utility generates, and `sid` is the system identifier for the database on which you want to suppress automatic fixups:

`DATABASE_LOGS_DIR``/`*`job-id`*`/prechecks/sid_checklist.cfg`

Open the `checklist.cfg` file with a text editor. The following is an example of the `checklist.cfg` file for the database with the SID `DB11204`:

```
[dbname]          [DB11204]
=========================================
[container]          [DB11204]
=========================================
[checkname]          AMD_EXISTS
[stage]              PRECHECKS
[fixup_available]    YES
[runfix]             YES
[severity]           WARNING
-------------------------------------------------

[checkname]          DEPEND_USR_TABLES
[stage]              POSTCHECKS
[fixup_available]    YES
[runfix]             YES
[severity]           WARNING
-------------------------------------------------

[checkname]          DICTIONARY_STATS
[stage]              PRECHECKS
[fixup_available]    YES
[runfix]             YES
[severity]           RECOMMEND
-------------------------------------------------

[checkname]          EM_PRESENT
[stage]              PRECHECKS
[fixup_available]    YES
[runfix]             YES
[severity]           WARNING
-------------------------------------------------
.
.
.
```

The file has a hierarchical structure. It starts with the database name, and the container name for which the entries of the `checklist.cfg` apply. The file contains a series of fixup checks that are applied automatically. For each entry, there are 5 relevant values parameters:

Parameter	Description
`[checkname]`	Name of the database check
`[stage]`	AutoUpgrade stage in which the check is performed. It can be either prechecks, or post checks.
`[fixup available]`	Availability of an automatic fixup. This parameter value is either `YES` (an automatic fixup is available), or `NO` (an automatic fixup is not available).

Parameter	Description
runfix	Run status for the fixup. This parameter takes one of two values: YES: Run the fixup. NO: Do not run the fixup.
severity	Class of severity of the issue that the automatic fixup addresses.

For each fixup that you want to perform manually, change the [runfix] parameter value from YES to NO.

Related Topics

- My Oracle Support Doc ID 2380601.1 "Database Preupgrade tool check list"

Local Configuration File Parameter Fixups Checklist Example

To include or exclude specific fixups for individual databases during upgrades, use the local configuration file checklist.

In this example, there is a local checklist file called sales4_checklist.cfg, which provides a preupgrade fixup checklist for the database sales4. A portion of the file contains the following settings:

```
[checkname] DICTIONARY_STATS
[stage] PRECHECKS
[fixup_available] YES
[runfix] YES
[severity] RECOMMEND
```

You can change the default fixup for DICTIONARY_STATS to exclude performing a fixup for the database sales4 by changing the runfix option for the check from YES to NO:

```
[checkname] DICTIONARY_STATS
[stage] PRECHECKS
[fixup_available] YES
[runfix] NO
[severity] RECOMMEND
```

Proper Management of AutoUpgrade Database Changes

AutoUpgrade is a powerful utility, which requires that you use it responsibly. Review and avoid using AutoUpgrade in ways that put the database at risk.

The following is a list of improper uses of AutoUpgrade, and ways of attempting to work around problems that result from these errors.

Breaking AutoUpgrade Resume Capability During Deployment

Problem Description::Using the `–clean_recovery_data` option prevents AutoUpgrade from resuming or restoring the database.

Workaround: Restore from a backup copy of the database.

Cause: Running the AutoUpgrade in deploy mode, and then interrupting its execution on any stage after the fixups are completed, and running the option `clean_recovery_data` before resuming and completing successfully an AutoUpgrade deploy command. For example:

```
java -jar autoupgrade.jar -config config.cfg -mode deploy
Ctrl+C //sample interruption
java -jar autoupgrade.jar -config config.cfg –clean_recovery_data
java -jar autoupgrade.jar -config config.cfg -mode deploy
```

Changing AutoUpgrade Global Log Directory During or After Deployment

Problem Description: If you change the global directory during or after running a deploy command, then the AutoUpgrade utility is unable to resume its pending work.

Workaround: Restore from a backup copy of the database.

Cause: The AutoUpgrade global logs directory also contains files used by the AutoUpgrade Utility to track the state of its operations. If you run the tool in deploy mode, and the deploy operation is stopped, and then rename or drop the global log directory, then the AutoUpgrade utility is unable to determine the state in which the deploy operation was stopped. As a result, when you restart AutoUpgrade, it begins the upgrade operation from the beginning, and the initial GRP is overwritten. You cannot use that GRP to restore the original database.

Use of Oracle Wallet With Credentials Not Set With AUTOLOGIN

Problem Description: You run the tool, and you have wallet credentials configured, but AutoUpgrade is unable to log in to the database.

Workaround: Create an Autologin Keystore, and configure the database with auto-login enabled.

At the time of this release. AutoUpgrade does not support the use of wallet credentials unless they are configured for automatic logins into the database

Related Topics

- Performing Operations That Require a Keystore Password

AutoUpgrade and Microsoft Windows ACLs and CLIs

When running AutoUpgrade on Microsoft Windows systems, Oracle recommends additional best practices with access control lists (ACLs) and command-line interfaces (CLIs).

AutoUpgrade and Access Control Lists (ACLs)

When you use AutoUpgrade on Windows systems, there are difficulties in setting up automated tools to work with Windows access control lists. Oracle strongly recommends that you complete the following best practice procedures:

- Review permissions for each of your target databases, and how these permissions relate to directories for these databases, such as the Oracle base directory, and the `oraInventory` files.
- Refer to the *Oracle Database Administrator's Reference* section on postinstallation configuration tasks for NTFS file systems.
- Review Microsoft's documentation regarding Windows PowerShell.
- Review the permissions for the groups `ORA_DBA`, `ORA_HOME_USERS`, and `ORA_ASM` groups. The `ORA_DBA` group only provides `SYSDBA` privileges to Oracle Database. The Oracle ASM management privileges are controlled by members of the group `ASM DISKGROUPS`.
- The `ORA_DBA` group member permissions to perform many administration tasks is limited, compared to the privileges available on `POSIX` systems. To enable AutoUpgrade to run as expected, Oracle recommends that the user account with Administrator rights on the Microsoft Windows server also manages Oracle base directory elements such as traces, listeners, and configuration.
- AutoUpgrade must be run using a command console (`CMD`) with administrative rights, and that console should be opened as the Oracle Installation User, or a user with similar privileges.
- Refer to My Oracle Support notes 1529702.1, and 1595375.1.

AutoUpgrade uses the following procedure with services running on the database:

- AutoUpgrade stops the services on the source database, and creates a temporary service on the target database Oracle home.

  If a restore is required, then the service in the target is dropped, and the service in the source is restarted.
- After Deploy Mode processing has completed successfully, the service in the target is dropped. At that point, it is the responsibility of the DBA for the upgraded Oracle Database to use `ORADIM` to create a service. Creating this service manually is required, because AutoUpgrade does not have the password to obtain permissions to create the ORADIM service.

AutoUpgrade and Windows Command-Line Interfaces

With command-line interfaces on Windows, applications can stop responding while waiting for a return character to be sent to the console. This behavior can affect the AutoUpgrade utility. The cause is a well-known Microsoft Windows console window characteristic related to the QEM (Quick Edit Mode). Even if you disable the Quick Edit console mode, the application can still encounter this behavior.

To avoid the program waiting for a response, press the enter key a few times after the application starts. Doing this provides the terminal input required to help the application proceed without awaiting a terminal response.

Related Topics

- *Oracle Database Administrator's Reference for Microsoft Windows*

- My Oracle Support Note 1529702.1
- My Oracle Support Note 1595375.1

4

Upgrading Oracle Database Manually Using Parallel Upgrade Utility

You can upgrade manually by using the Parallel Upgrade Utility command-line option.

Starting with Oracle Database 19c, Database Upgrade Assistant (DBUA) is replaced by the AutoUpgrade utility.

> **Caution:**
>
> If you retain the old Oracle Database software, then never start the upgraded database with the old Oracle Database software. Only start the database with the executables in the new Oracle Database installation.

- Upgrading Manually with Parallel Upgrade Utility
  To run upgrades with scripts that you run and manage manually, you can use the Parallel Upgrade Utility (`catctl.pl`).
- Manual Upgrade Scenarios for Multitenant Architecture Oracle Databases
  To prepare for manual upgrades, review the manual upgrade scenarios and procedures for Oracle Database deployed with multitenant architecture.
- Example of Manual Upgrade of Windows Non-CDB Oracle Database
  These examples show the steps to complete preupgrade checks, upgrade, and postupgrade checks for an Oracle Database 11g Release 2 (11.2.0.4) upgrade to Oracle Database 19c.
- About Dbupgrade Scripts and catupgrd.sql in Earlier Releases of Oracle Database
  The function of the `catupgrd.sql` script is replaced by the Parallel Upgrade Utility, `catctl.pl`, and the `dbupgrade` and `dbupgrade.cmd` scripts.
- About Transporting and Upgrading a Database (Full Transportable Export/Import)
  You can use file-based or nonfile-based modes for transporting data.
- Upgrade Scenarios for Non-CDB Oracle Databases
  Review these topics to understand the upgrade scenarios and procedures for non-CDB Oracle Databases
- Rerunning Upgrades for Oracle Database
  Use these options to rerun upgrades.

Rerunning Upgrades for Oracle Database

Use these options to rerun upgrades.

- About Rerunning Upgrades for Oracle Database
  Oracle provides the features listed here to rerun or restart Oracle Database upgrades, including after failed phases.

- Rerunning Upgrades with the Upgrade (catctl.pl) Script
  You can fix upgrade issues and then rerun the upgrade with the `catctl.pl` script, or the `dbupgrade` shell command.
- Options for Rerunning the Upgrade for Multitenant Databases (CDBs)
  There are four options you can use to rerun upgrades on multitenant database architecture (CDBs) for Oracle Database 12c release 2 (12.2).
- Restarting the Upgrade from a Specific Phase that Failed Using -p
  Use this option to complete an upgrade after fixing errors.

About Rerunning Upgrades for Oracle Database

Oracle provides the features listed here to rerun or restart Oracle Database upgrades, including after failed phases.

Parallel Upgrade Utility and Restarts or Reruns

You can re-run or restart Oracle Database upgrade phases by using either Database Upgrade Assistant (DBUA), or by using the Parallel Upgrade Utility (`catctl.pl`) script. You can also run commands on PDBs that failed to upgrade in an initial attempt, so that you can complete the upgrade.

Parallel Upgrade Utility Resume Option

Oracle Database 12*c* release 2 (12.2) includes a new Resume option for Parallel Upgrade Utility. This option is available for both CDBs and Non-CDBs. You are not required to identify failed or incomplete phases when you rerun or restart the upgrade. When you use the Parallel Upgrade Utility using the resume option (`-R`), the utility automatically detects phases from the previous upgrade that are not completed successfully. The Parallel Upgrade Utility then reruns or restarts just these phases that did not complete successfully, so that the upgrade is completed. Bypassing steps that already completed successfully reduces the amount of time it takes to rerun the upgrade.

To use the Resume option, run the Parallel Upgrade Utility using the -R parameter. For example:

```
$ORACLE_HOME/perl/bin/perl catctl.pl -L plist.txt -n 4 -N 2 -R -l
$ORACLE_HOME/cfgtoollogs catupgrd.sql
```

You can rerun the entire upgrade at any time, regardless of which phase you encountered a failure in your upgrade. If you plan to rerun the entire upgrade, instead of rerunning only failed phases, then run the Parallel Upgrade Utility without using the Resume (`-R`) option.

Rerunning Upgrades with the Upgrade (catctl.pl) Script

You can fix upgrade issues and then rerun the upgrade with the `catctl.pl` script, or the `dbupgrade` shell command.

> **Note:**
>
> Starting with Oracle Database 12c, Release 1 (12.1), non-CDB architecture is deprecated. It can be desupported in a future release.

1. Shut down the database. For a non-CDB and a CDB, the syntax is the same.

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

2. Restart the database in `UPGRADE` mode.

   For a non-CDB:

   ```
   SQL> STARTUP UPGRADE
   ```

   For a CDB:

   ```
   SQL> STARTUP UPGRADE
   SQL> alter pluggable database all open upgrade;
   ```

3. Rerun the Parallel Upgrade utility (`catctl.pl`, or `dbupgrade` shell command).

   You can rerun the Parallel Upgrade Utility as many times as necessary.

   With CDBs, you can use the Resume option (`-R`) to rerun the Parallel Upgrade Utility. The script resumes the upgrades from failed phases.

   For example:

   ```
   $ORACLE_HOME/perl/bin/perl catctl.pl -n 4 -R -l $ORACLE_HOME/
   cfgtoollogs catupgrd.sql
   ```

   You can also provide the name of one or more specific PDBs on which you want to rerun the upgrade.

   For example, this command reruns the upgrade on the PDB named `cdb1_pdb1`:

   ```
   $ORACLE_HOME/perl/bin/perl catctl.pl -n 4 -R -l $ORACLE_HOME/
   cfgtoollogs -c 'cdb1_pdb1' catupgrd.sql
   ```

   You can use the `dbupgrade` shell command to run the same commands:

   ```
   dbupgrade -n 4 -R -l $ORACLE_HOME/diagnostics
   dbupgrade -n 4 -R -l $ORACLE_HOME/diagnostics -c 'cdb1_pdb1'
   ```

4. Run `utlusts.sql`, the Post-Upgrade Status Tool, which provides a summary of the status of the upgrade in the spool log. You can run `utlusts.sql` any time before or after you complete the upgrade, but not during the upgrade.

   In a non-CDB:

   ```
   SQL> @$ORACLE_HOME/rdbms/admin/utlusts.sql
   ```

In a CDB:

```
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlu122s -d '''.''' 
utlusts.sql
```

If the `utlusts.sql` script returns errors or shows components that are not `VALID` or not the most recent release, then follow troubleshooting procedures for more information.

5. Run `utlrp.sql` to recompile any remaining stored PL/SQL and Java code.

```
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlrp -d '''.''' 
utlrp.sql
```

The script generates the log file `utlrp0.log` , which shows the results of the re-compilations.

Use the following SQL commands to verify that all expected packages and classes are valid,

In a single PDB (`cdb1_pdb1` in this example), open the PDB in normal mode as follows:

```
alter pluggable database cdb1_pdb1 open;
```

Run `catcon.pl` to start `utlrp.sql` in the PDB to recompile any remaining stored PL/SQL and Java code. Use the following syntax:

```
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlrp -d '''.''' -c 
'cdb1_pdb1'
 utlrp.sql
```

In a non-CDB:

```
SQL> SELECT count(*) FROM dba_invalid_objects;
SQL> SELECT distinct object_name FROM dba_invalid_objects;
```

In an entire CDB:

```
SQL> ALTER SESSION SET CONTAINER = "CDB$ROOT"
SQL> SELECT count(*) FROM dba_invalid_objects;
SQL> SELECT distinct object_name FROM dba_invalid_objects;
SQL> ALTER SESSION SET CONTAINER = "PDB$SEED"
SQL> SELECT count(*) FROM dba_invalid_objects;
SQL> SELECT distinct object_name FROM dba_invalid_objects;
SQL> ALTER SESSION SET CONTAINER = "cdb1_pdb1"
SQL> SELECT count(*) FROM dba_invalid_objects;
SQL> SELECT distinct object_name FROM dba_invalid_objects;
```

6. Run `utlusts.sql` again to verify that all issues have been fixed.

In a non-CDB:

```
SQL> @$ORACLE_HOME/rdbms/admin/utlusts.sql
```

In a CDB:

```
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlu122s -d '''.''' utlusts.sql
```

7. Exit SQL*Plus.
8. If you are upgrading a cluster database from Release 11.2, then upgrade the database configuration in Oracle Clusterware using the following command syntax, where *db-unique-name* is the database name assigned to it (not the instance name), and *Oracle_home* is the Oracle home location in which the database is being upgraded.

```
$ srvctl upgrade database -d db-unique-name -o Oracle_home
```

Your database is now upgraded. You are ready to complete post-upgrade tasks for Oracle Database.

Related Topics

- Options for Rerunning the Upgrade for a Multitenant Database (CDB)
  There are four options you can use to rerun upgrades on multitenant database architecture (CDBs) for Oracle Database 12c release 2 (12.2).
- Troubleshooting the Upgrade for Oracle Database
  Use these troubleshooting tips to address errors or issues that you may encounter while upgrading your database.
- Post-Upgrade Tasks for Oracle Database
  After you upgrade Oracle Database, complete required post-upgrade tasks, and consider recommendations for the new release.

Options for Rerunning the Upgrade for Multitenant Databases (CDBs)

There are four options you can use to rerun upgrades on multitenant database architecture (CDBs) for Oracle Database 12c release 2 (12.2).

- Rerun the Entire Upgrade for the CDB
  If several different issues occur during the first upgrade attempt, then use this procedure to re-run the entire upgrade.
- Rerun the Upgrade Only on Specified PDBs
  You can rerun upgrades on specified multitenant containers by running the Parallel Upgrade Utility with either the Resume option (`-R`), or with the exclusion list option (`-C`).
- Rerun the Upgrade While Other PDBs Are Online
  You can rerun PDB upgrades by using the Parallel Upgrade Utility Resume option, or by explicitly including or excluding online PDBs using with inclusion or exclusion lists.
- Rerun the Upgrade Using an Inclusion List to Specify a CDB or PDBs
  Use this example as a model for rerunning an upgrade on a PDB by using an inclusion list.

Rerun the Entire Upgrade for the CDB

If several different issues occur during the first upgrade attempt, then use this procedure to re-run the entire upgrade.

This example demonstrates running the upgrade again on the `CDB$ROOT`, `PDB$SEED` and all PDBs after correcting for a problem occurring during the initial upgrade attempt, such as running out of shared pool.

1. Start the upgrade again. For example:

```
SQL> startup upgrade;
alter pluggable database all open upgrade;
```

2. Run the Parallel Upgrade Utility (`catctl.pl`, or the `dbupgrade` shell script. For example:

```
cd $ORACLE_HOME/bin/
./dbupgrade -d $ORACLE_HOME/rdbms/admin -l $ORACLE_HOME/rdbms/log
```

   The upgrade runs again on all the containers, including `CDB$ROOT` , `PDB$SEED`, and all PDBs in the CDB.

Rerun the Upgrade Only on Specified PDBs

You can rerun upgrades on specified multitenant containers by running the Parallel Upgrade Utility with either the Resume option (`-R`), or with the exclusion list option (`-C`).

In both the examples that follow, the multitenant container database contains five PDBs. All upgrades ran successfully except for `CDB1_PDB1` and `CDB1_PDB2`, which failed with an upgrade error. To run the upgrade on these two containers, you shut down the entire multitenant database and restart only the PDBs you want to upgrade.

> **Note:**
>
> Parallel Upgrade Utility parameters are case-sensitive.

Example 4-1 Rerunning Upgrades With the Resume Option

You can use the Parallel Upgrade Utility Resume parameter option `-R` to rerun the upgrade only on one or more multitenant containers (CDBs).

In the following example, the upgrade script detects that it should run on `CDB1_PDB1` and `CDB1_PDB2` containers only.

1. Shut down the multitenant database, start up the database in upgrade mode, and then start up the PDBs on which the upgrade did not complete. For example:

```
SQL> shutdown immediate;
      startup upgrade;
```

```
      alter pluggable database CDB1_PDB1 open upgrade;
      alter pluggable database CDB1_PDB2 open upgrade;
```

2. Show the CDB and PDB status:

```
 SQL> show pdbs

       CON_ID CON_NAME                       OPEN MODE  RESTRICTED
       ---------- ------------------------------ ----------
----------
           2 PDB$SEED                       MIGRATE    YES
           3 CDB1_PDB1                      MIGRATE    YES
           4 CDB1_PDB2                      MIGRATE    YES
           5 CDB1_PDB3                      MOUNTED
           6 CDB1_PDB4                      MOUNTED
           7 CDB1_PDB5                      MOUNTED
```

3. Rerun the upgrade. The upgrade automatically detects from the previous upgrade logs that CDB$ROOT and PDB$SEED are upgraded successfully. The upgrade bypasses CDB$ROOT and PDB$SEED, and only runs on CDB1_PDB1 and CDB_PDB2. The command example here is for Linux/UNIX systems:

```
cd $ORACLE_HOME/bin
./dbupgrade -d $ORACLE_HOME/rdbms/admin -l $ORACLE_HOME/cfgtoollogs
-R
```

The Parallel Upgrade Utility completes the upgrade on `CDB1_PDB1` and `CDB1_PDB2`.

Example 4-2 Rerunning Upgrades With an Exclusion List

An exclusion list contains containers that you do not want to upgrade. An exclusion list uses the Parallel Upgrade Utility `-C` parameter option. Run the Parallel Upgrade utility by changing directory to *`Oracle_home`*`/rdbms/admin/` and running the utility in Perl using `catctl.pl`, or by changing directory to *`Oracle_home`*`/bin` and running the command-line script, `dbupgrade -C`. This method is useful when you have many PDBs on which you want to rerun the upgrade.

In this following example, you provide an exclusion list to exclude the upgrade script from running on containers where you do not require it to run.

1. Shut down the multitenant database, start up the database in upgrade mode, and then start up the PDBs on which the upgrade did not complete. For example:

```
SQL> shutdown immediate;
      startup upgrade;
      alter pluggable database CDB1_PDB1 open upgrade;
      alter pluggable database CDB1_PDB2 open upgrade;
```

2. Show the CDB and PDB status:

```
 SQL> show pdbs

       CON_ID CON_NAME                       OPEN MODE  RESTRICTED
       ---------- ------------------------------ ----------
----------
           2 PDB$SEED                       MIGRATE    YES
```

```
3 CDB1_PDB1                    MIGRATE    YES
4 CDB1_PDB2                    MIGRATE    YES
5 CDB1_PDB3                    MOUNTED
6 CDB1_PDB4                    MOUNTED
7 CDB1_PDB5                    MOUNTED
```

3. Rerun the upgrade, excluding `CDB$ROOT` and `PDB$SEED` from the upgrade in a space-delimited exclusion list that you specify with single quote marks. The command example here is for Linux/UNIX systems:

   ```
   $ORACLE_HOME/bin/dbupgrade -d $ORACLE_HOME/rdbms/admin -l
   $ORACLE_HOME/rdbms/log -C 'CDB$ROOT PDB$SEED'
   ```

The upgrade reruns, and completes on `CDB1_PDB1` and `CDB1_PDB2`.

> **Note:**
>
> For Windows, you must specify the `-C` option exclusion list by using with double quote marks to specify the exclusion list. For example:
>
> ```
> ... -C "CDB$ROOT PDB$SEED"
> ```

Rerun the Upgrade While Other PDBs Are Online

You can rerun PDB upgrades by using the Parallel Upgrade Utility Resume option, or by explicitly including or excluding online PDBs using with inclusion or exclusion lists.

Use these examples as a model for running upgrades on PDBs, where you want to rerun upgrades on some PDBs while other PDBs are open.

In the examples, the upgrade failed in containers `CDB1_PDB1` and `CDB1_PDB2`, but succeeded in containers `CDB1_PDB3`, `CDB1_PDB4`, and `CDB1_PDB5`.

You start up `CDB$ROOT` in normal mode. You find that the following containers are online: `CDB1_PDB3`, `CDB1_PDB4`, and `CDB1_PDB5` . You review the upgrade logs for `CDB1_PDB3`, `CDB1_PDB4`, and `CDB1_PDB5` and bring these containers online.

Example 4-3 Rerunning Upgrades on PDBs Using the Resume Option

The following example shows how to complete the upgrade for `CDB1_PDB1` and `CDB1_PDB2` by using the Parallel Upgrade Utility Resume option. The Resume option excludes PDBs that are already upgraded:

1. Bring up `CDB$ROOT` in normal mode, and open `CDB1_PDB1` and `CDB1_PDB2` in upgrade mode. `CDB1_PDB3`, `CDB1_PDB4`, `CDB1_PDB5` are in normal mode. For example:

   ```
   SQL> startup;
         alter pluggable database CDB1_PDB1 open upgrade;
         alter pluggable database CDB1_PDB2 open upgrade;
         alter pluggable database cdb1_pdb3 open;
         alter pluggable database cdb1_pdb4 open;
         alter pluggable database cdb1_pdb5 open;
   ```

2. Use the SQL command `show pdbs` to show the status of PDBs. For example:

```
SQL> show pdbs

        CON_ID CON_NAME                       OPEN MODE  RESTRICTED
        ---------- ------------------------------ ---------- ----------
             2 PDB$SEED                       READ ONLY  NO
             3 CDB1_PDB1                      MIGRATE    YES
             4 CDB1_PDB2                      MIGRATE    YES
             5 CDB1_PDB3                      READ WRITE NO
             6 CDB1_PDB4                      READ WRITE NO
             7 CDB1_PDB5                      READ WRITE NO
```

3. Rerun the upgrade, excluding `CDB$ROOT` from the upgrade, using the Parallel Upgrade Utility command-line script `dbupgrade`.

```
cd $ORACLE_HOME/bin
./dbupgrade -d $ORACLE_HOME/rdbms/admin/ -l $ORACLE_HOME/
cfgtoollogs -R
```

Because you run the upgrade using the Resume option, the Parallel Upgrade Utility checks the logs and identifies that `CDB1_PDB1` and `CDB1_PDB2` are the only two containers in `CDB1` that are not upgraded. The upgrade script runs on just those two PDBs. The upgrade does not rerun on `PDB$SEED`, `CDB$ROOT`, `CDB_PDB3`, `CDB_PDB4`, and `CDB_PDB5`.

Example 4-4 Rerunning Upgrades on PDBs Using Exclusion Lists

The following example shows how to complete the upgrade for `CDB1_PDB1` and `CDB1_PDB2` by using an exclusion list to exclude the PDBs on which you do not want the upgrade script to run:

1. Bring up `CDB$ROOT` in normal mode, and open `CDB1_PDB1` and `CDB1_PDB2` in upgrade mode. `CDB1_PDB3`, `CDB1_PDB4`, `CDB1_PDB5` are in normal mode. For example:

```
SQL> startup;
      alter pluggable database CDB1_PDB1 open upgrade;
      alter pluggable database CDB1_PDB2 open upgrade;
      alter pluggable database cdb1_pdb3 open;
      alter pluggable database cdb1_pdb4 open;
      alter pluggable database cdb1_pdb5 open;
```

2. Use the SQL command `show pdbs` to show the status of PDBs. For example:

```
SQL> show pdbs

        CON_ID CON_NAME                       OPEN MODE  RESTRICTED
        ---------- ------------------------------ ---------- ----------
             2 PDB$SEED                       READ ONLY  NO
             3 CDB1_PDB1                      MIGRATE    YES
             4 CDB1_PDB2                      MIGRATE    YES
             5 CDB1_PDB3                      READ WRITE NO
             6 CDB1_PDB4                      READ WRITE NO
             7 CDB1_PDB5                      READ WRITE NO
```

3. Rerun the upgrade, excluding CDB$ROOT from the upgrade, using the Parallel Upgrade Utility command-line script dbupgrade:

```
cd $ORACLE_HOME/bin
./dbupgrade -d $ORACLE_HOME/rdbms/admin -l $ORACLE_HOME/cfgtoollogs
-R -C 'CDB$ROOT'
```

   The Parallel Upgrade Utility runs with the Resume option (-R), and identifies from the logs that CDB1_PDB1 and

   CDB1_PDB2 have not completed the upgrade. Because the Parallel Upgrade Utility runs with the Exclude option (-C), and you specify that CDB$ROOT is excluded, the upgrade script is also explicitly excluded from running on CDB$ROOT.

   For Windows, when you run the Parallel Upgrade Utility with the Exclude option (-C), you must specify the targets -C option using double quotes around the CDB root name and PDB seed name. For example:

```
. . . -C "CDB$ROOT PDB$SEED"
```

4. The upgrade reruns and completes on CDB1_PDB1 and CDB1_PDB2.

Rerun the Upgrade Using an Inclusion List to Specify a CDB or PDBs

Use this example as a model for rerunning an upgrade on a PDB by using an inclusion list.

You can use an inclusion list to specify a list of CDBs and PDBs where you want to re-run an upgrade, and exclude nodes not on the inclusion list. Specify the inclusion list by running the Parallel Upgrade Utility with the inclusion option (-c), followed by a space-delimited list designated by single quotes of the containers that you want to upgrade.

In the examples, the upgrade failed in containers CDB1_PDB1 and CDB1_PDB2, but succeeded in containers CDB1_PDB3, CDB1_PDB4, and CDB1_PDB5. You start up CDB$ROOT in normal mode. You find that the following containers are online: CDB1_PDB3, CDB1_PDB4, and CDB1_PDB5 . You review the upgrade logs for CDB1_PDB3, CDB1_PDB4, and CDB1_PDB5 and bring these containers online.

Example 4-5 Rerunning Upgrades on PDBs Using an Inclusion List

For example:

1. Bring up CDB$ROOT in normal mode, and open CDB1_PDB1 and CDB1_PDB2 in upgrade mode. CDB1_PDB3, CDB1_PDB4, CDB1_PDB5 are in normal mode. For example:

```
SQL> startup;
      alter pluggable database CDB1_PDB1 open upgrade;
      alter pluggable database CDB1_PDB2 open upgrade;
      alter pluggable database cdb1_pdb3 open;
      alter pluggable database cdb1_pdb4 open;
      alter pluggable database cdb1_pdb5 open;
```

2. Use the SQL command show pdbs to show the status of PDBs. For example:

```
SQL> show pdbs
```

```
CON_ID CON_NAME                       OPEN MODE  RESTRICTED
------ ------------------------------ ---------- ----------
     2 PDB$SEED                       READ ONLY  NO
     3 CDB1_PDB1                      MIGRATE    YES
     4 CDB1_PDB2                      MIGRATE    YES
     5 CDB1_PDB3                      READ WRITE NO
     6 CDB1_PDB4                      READ WRITE NO
     7 CDB1_PDB5                      READ WRITE NO
```

3. Rerun the Parallel Upgrade Utility with the inclusion (-c) option, followed by a space-delimited inclusion list that you specify with single quote marks. This option runs the upgrade only on the PDBs that you list in the inclusion list.

   For example:

   ```
   cd $ORACLE_HOME/bin
   ./dbupgrade -d $ORACLE_HOME/rdbms/admin -l ORACLE_HOME/cfgtoollogs
   -R -c 'CDB1_PDB1 CDB1_PDB2'
   ```

   For Windows, when you run the Parallel Upgrade Utility with the inclusion option, you must specify the `-c` option targets by using double quotes around the inclusion list. For example:

   ```
   . . . -C "CDB1_PDB1 CDB1_PDB2"
   ```

4. The upgrade reruns and completes on `CDB1_PDB1` and `CDB1_PDB2`.

Restarting the Upgrade from a Specific Phase that Failed Using -p

Use this option to complete an upgrade after fixing errors.

You can run the Parallel Upgrade Utility (`catctl.pl`, or the shell scripts `dbupgrade` or `dbupgrade.cmd`) with the `-p` option to rerun an upgrade and skip upgrade phases that already have run successfully. You can also rerun the upgrade on one phase to test the fix for failed phases.

To determine the phase number to restart, examine the upgrade log to identify where the first error occurred, and in what phase. You can then fix the cause of the error, and test the fix or rerun the upgrade to completion.

- Reviewing CDB Log Files for Failed Phases
  Identify your log file location, and review the CDB and PDB log files.
- Procedure for Finding and Restarting Multitenant Upgrades from a Failed Phase
  To restart a multitenant upgrade from a failed phase, first identify which PDB created the error and then search its appropriate log file for the error.

Reviewing CDB Log Files for Failed Phases

Identify your log file location, and review the CDB and PDB log files.

The location of the Automatic Diagnostic Repository (ADR) and the diagnostic log files created by the upgrade scripts can vary, depending on your environment variables and parameter settings.

You can set log file paths when you run the Parallel Upgrade Utility (`catctl`) by setting the `-l` option to define a log file path.

Log files for CDB$ROOT (CDBs) can span from `catupgrd0...catupgrd7.log`. Log files for pluggable databases (PDBs) are identified by the PDB container name (*dbname*), and span from `catupgrdp`*dbname*`0...catupgrdp`*dbname*`7.log`.

Procedure for Finding and Restarting Multitenant Upgrades from a Failed Phase

To restart a multitenant upgrade from a failed phase, first identify which PDB created the error and then search its appropriate log file for the error.

To identify the PDB that caused a multitenant upgrade failure, look at the upgrade summary report, or review catupgrd0.log; this log contains the upgrade summary report at the end of the file.

Use this procedure to check each log file looking for errors.

1. Locate log files with errors.

   For example:

   Linux and Unix

   ```
   $ grep -i 'error at line' catupgrd*.log
   ```

   Windows

   ```
   C:\> find /I "error at line" catupgrd*.log
   ```

   The `grep` or `find` command displays the filenames of log files in which an error is found.

2. Check each log file that has an error and identify where the first error occurred. Use the text editor of your choice to review each log file. Search for the first occurrence of the phrase `error at line`'. When you find the phrase, then search backwards from the error (reverse search) for `PHASE_TIME___START`.

   For example:

   ```
   PHASE_TIME___START 15 15-01-16 08:49:41
   ```

   The number after `PHASE_TIME___START` is the phase number where the error has occurred. In this example, the phase number is 15.

   Each log file can have an error in it. Repeat checking for the phrase `PHASE_TIME___START`, and identify the phase number with errors for each log file that contains an error, and identify the log file that contains the lowest phase number.

   The log file that contains the lowest phase number is restart phase number, which is the phase number from which you restart the upgrade.

   For example:

   ```
   catupgrd0.log error occurred in phase 15:

   PHASE_TIME___START 15 15-01-16 08:49:41
   ```

```
catupgrd1.log error occurred in phase 19:

PHASE_TIME___START 19 15-01-16 08:50:01
```

In this example, the restart phase number is 15. Ensure that you identify the first error seen in all the log files, so that you can restart the upgrade from that phase.

3. Restart the upgrade from the failed phase by changing directory to the running the Parallel Upgrade Utility (`catctl.pl`, which you can run from the command line using `dbupgrade` on Linux and Unix, and `dbupgrade.cmd` on Windows). Use the `-p` flag to indicate that you want to restart the upgrade from a phase, and provide the restart phase number. In multitenant databases, also use the `-c` flag using the syntax `-c '`*`PDBname`*`'`, where *`PDBname`* is the name of the PDB where the failure occurred.

   For example:

   Non-CDB Oracle Database on a Linux or UNIX system:

   ```
   cd $ORACLE_HOME/bin
   dbupgrade -p 15
   ```

   PDB in a multitenant Oracle Database (CDB) on a Windows system:

   ```
   cd $ORACLE_HOME/bin
   dbupgrade -p 15 -c 'PDB1'
   ```

   In both examples, the upgrade is restarted from phase 15, identified with the `-p` flag. In the multitenant example, the PDB with the error is identified with the `-c` flag.

   In these examples, the upgrade starts from phase 15 and runs to the end of the upgrade.

4. (Optional) You can also run the phase that contained an error by specifying a stop phase, using the `-P` flag. Using a stop phase allows the upgrade to just rerun that phase in which the error occurred. You can determine quickly if the error is fixed by running it on the phase with the error, without running the entire upgrade.

   For example, using the Perl script Parallel Upgrade Utility command option:

   ```
   cd $ORACLE_HOME/rdbms/admin
   $ORACLE_HOME/perl/bin/perl catctl.pl -p 15 -P 15 -c 'PDB1'
   ```

   After you confirm that the error is fixed in the phase with the error, you can then resume the upgrade after that phase.

   For example, if you have confirmed that the error in phase 15 of your multitenant database upgrade of PDB1 is fixed, then you can use the following command on Linux and Unix systems to continue the upgrade at phase 16:

   ```
   cd $ORACLE_HOME/bin dbupgrade -p 16 -c 'PDB1'
   ```

Upgrading Manually with Parallel Upgrade Utility

To run upgrades with scripts that you run and manage manually, you can use the Parallel Upgrade Utility (`catctl.pl`).

- About the Parallel Upgrade Utility for Oracle Database (CATCTL.PL and DBUPGRADE)
  The Parallel Upgrade Utility (`catctl.pl`, and the `dbupgrade` script) enable you to upgrade simultaneously components that do not require upgrades to occur in a specific order.
- General Steps for Running the Parallel Upgrade Utility
  Review to obtain an overview of how to use the Parallel Upgrade Utility for Oracle Database.
- Parallel Upgrade Utility (catctl.pl) Parameters
  Control how the Parallel Upgrade Utility (`catctl.pl`) runs. You can also use these arguments to run the `dbupgrade` shell command.
- Example of Using the Parallel Upgrade Utility
  Use this example to understand how you can run the parallel upgrade utility manually to perform upgrades.

About the Parallel Upgrade Utility for Oracle Database (CATCTL.PL and DBUPGRADE)

The Parallel Upgrade Utility (`catctl.pl`, and the `dbupgrade` script) enable you to upgrade simultaneously components that do not require upgrades to occur in a specific order.

Oracle Database 12*c* release 1 (12.1) introduced the Parallel Upgrade Utility, `catctl.pl`. This utility reduces the total amount of time it takes to perform an upgrade by loading the database dictionary in parallel, and by using multiple SQL processes to upgrade the database. Performing parallel upgrades of components enables you to take advantage of your CPU capacity. Oracle continues to make improvements to the upgrade process to simplify both manual upgrades, and upgrades performed with the Database Upgrade Assistant (DBUA). DBUA and the manual upgrade procedures take advantage of the new Parallel Upgrade Utility.

You can run a shell command, `dbupgrade`, which starts up `catctl.pl` from the command line, instead of requiring you to run it from Perl.

The `dbupgrade` shell command is located in the file path `$ORACLE_HOME/bin` on Linux and UNIX, and `%ORACLE_HOME%\bin` on Windows. You can provide any command arguments that are valid for `catctl.pl` to the shell command. Either run the command directly from the new Oracle home path, or set a user environment variable to point to the file path.

For example:

Running with default values:

```
$ ./dbupgrade
```

Running to specify a log directory placed in `/tmp`:

```
$ ./dbupgrade -l /tmp
```

You can also run the Parallel Upgrade Utility using priority lists. For example:

```
$ ./dbupgrade -L priority_list_name
```

When you use a priority list, you can include or exclude a specific list of PDBs in your upgrade.

You can also run the Parallel Upgrade Utility using priority emulation, so that you can see how the priority list is read and carried out, without actually performing the upgrade. For example:

```
$ ./dbupgrade -E
```

Related Topics

- Example of Testing Upgrades Using Priority List Emulation

General Steps for Running the Parallel Upgrade Utility

Review to obtain an overview of how to use the Parallel Upgrade Utility for Oracle Database.

The Parallel Upgrade Utility (`catctl.pl`, which you can start with the shell command `dbupgrade`) loads the data dictionary and components in parallel. Loading in parallel reduces the overall upgrade time. Before running the Parallel Upgrade Utility, follow the procedures for backing up your database that you normally do before upgrading. Also, as a prerequisite, you must run the Pre-Upgrade Information Tool to identify any problems that a database administrator must address before the upgrade proceeds.

The general steps for upgrading your database with the Parallel Upgrade Utility are as follows:

1. Back up your current database.
2. Install the Oracle Database software for the new release.
3. Ensure that the Pre-Upgrade Information Tool (`preupgrade.jar`) has run on the source database, and that any issues reported by the tool are addressed.
4. Shut down your current database.
5. Set up the new Oracle home environment to access the new release database software, and then start SQL*Plus from the directory *`ORACLE_HOME`*`/rdbms/admin`.
6. Log in to a user account with `SYSDBA` system privileges, and connect to the database that you want to upgrade:

   ```
   CONNECT / AS SYSDBA
   ```

7. Start the database in upgrade mode. Use the command for your configuration type.

Multitenant container database (CDB):

```
SQL> startup upgrade;
SQL> alter pluggable database all open upgrade;
```

Non-CDB:

```
SQL> startup upgrade
```

> **Note:**
>
> The UPGRADE keyword performs operations that prepare the environment for the upgrade.

You can be required to use the `PFILE` option in your startup command to specify the location of your initialization parameter file.

When you start the database in upgrade mode, only queries on fixed views execute without errors until after the `catctl.pl` script is run. Before you run `catctl.pl`, you receive an error if you try to use PL/SQL, or if you try to run queries on any other view.

If errors appear listing desupported initialization parameters, then make a note of the desupported initialization parameters, and continue with the upgrade. Remove the desupported initialization parameters the next time you shut down the database.

8. Exit SQL*Plus.
9. Run the Parallel Upgrade Utility from the new Oracle home.

   You can run the utility as a shell command (`dbupgrade` on Linux and Unix, and `dbupgrade.cmd` on Microsoft Windows) or you can run it as a Perl command (`catctl.pl`).

   For example, on Linux and Unix:

   ```
   cd $ORACLE_HOME/bin
   ./dbupgrade
   ```

   For example, on Microsoft Windows:

   ```
   cd %ORACLE_HOME%\bin
   dbupgrade
   ```

   The Parallel Upgrade Utility starts the upgrade process.

> **Note:**
>
> The Parallel Upgrade Utility uses other files to carry out the upgrade. On Linux and Unix systems, these files include `catconst.pm`, `catcom.pm`, `sqlpatch`, `sqlpatch.pl` or `sqlpatch.pm`, and `orahome` on Linux/UNIX systems. On Windows systems, these files include `orahome.exe`. Do not change or remove these files.

Related Topics

- Specifying Initialization Parameters at Startup

Parallel Upgrade Utility (catctl.pl) Parameters

Control how the Parallel Upgrade Utility (`catctl.pl`) runs. You can also use these arguments to run the `dbupgrade` shell command.

> **Note:**
>
> The shell command utility `dbupgrade` starts `catctl.pl`. The dbupgrade utility resides in the `ORACLE_HOME/bin` directory. You can use the shell command utility to start the Parallel Upgrade Utility at the command prompt. You can either run the utility using default values, or you can use `catctl.pl` input parameters to specify Parallel Upgrade Utility arguments.

Table 4-1 Parallel Upgrade Utility (catctl.pl) Parameters

Parameter	Description

`-c`	Specifies a space-delimited inclusion list for PDBs that you want to upgrade. For example, in an Oracle Multitenant deployment with PDB1, PDB2, PDB3, and PDB4, include PDB1 and PDB2, but exclude the PDBs not named. PDB 1 and PDB 2 are upgraded, but PDB 3 and PDB4 are not upgraded.
	Linux and UNIX (use single quotes):
	`-c 'PDB1 PDB2'`
	Windows (use double quotes):
	`-c "PDB1 PDB2"`

Table 4-1 (Cont.) Parallel Upgrade Utility (catctl.pl) Parameters

Parameter	Description

`-C`	Specifies a space-delimited exclusion list for PDBs that you want to upgrade. For example, in an Oracle Multitenant deployment with PDB1, PDB2, PDB3, and PDB4, you can use an exclusion list to exclude PDB1 and PDB2, but include the PDBs not named. PDB1 and PDB2 are not upgraded, but PDB3 and PDB4 are upgraded.
	Linux and UNIX (use single quotes):
	`-C 'PDB1 PDB2'`
	Windows (use double quotes):
	`-C "PDB1 PDB2"`
	Note: `-c` and `-C` are mutually exclusive.
	`-C 'CATCTL_LISTONLY'` is an option that specifies that the Parallel Upgrade Utility processes only the PDBs in a priority list. Use this option with the `-L` parameter, specifying a list.

`-d`	Specifies the location of the directory containing the files that you want processed.
`-e`	Sets echo OFF while running the scripts. The default is echo ON.
`-E`	Enables you to run an upgrade emulation. You can use the `-E` parameter to run the Parallel Upgrade Utility in emulation mode to test how priority lists run, or to test how other upgrade parameter selections are carried out during an upgrade. For example, you can run an upgrade emulation to obtain more information about how the resource allocation choices you make using the `-n` and `-N` parameters are carried out. To carry out an upgrade emulation, complete all upgrade preparations before you run the Parallel Upgrade Utility, and then run the command using `-E`. When you run the Parallel Upgrade Utility with the `-E` parameter, and call a priority list as part of the command using the `-L` parameter, the utility writes the upgrade order to the file `catctl_priority_run.lst`. This list is placed in the file path that you specify by the `-l` parameter, or in the default log file area if you do not specify a different output file path.
`-F`	Forces a cleanup of previous upgrade errors. Non-CDB databases require only the `-F` parameter. For CDBs, use this option with a space-delimited inclusion list, which you specify with `-C`.
`-i`	Specifies an identifier to use when creating spool log files.

Table 4-1 (Cont.) Parallel Upgrade Utility (catctl.pl) Parameters

Parameter	Description
`-l`	Specifies the location for the directory to use for spool log files. The default location is *`Oracle_base`*`/cfgtoollogs/dbname/upgrade`*`datetime`*. The *`date`* and *`time`* strings are in the character string format *YYYYMMDDHHMMSC*, in which *YYYY* designates the year, *MM* designates the month, *DD* designates the day, *HH* designates the hour, *MM* designates the minute, and *SC* designates the second. Oracle strongly recommends that you do not write log files to the `/admin` directory.
`-L`	Upgrades PDBs using a priority list during an Oracle Database upgrade, and specifies the priority list name. The priority list updates priority status in the database during upgrade. This priority listing is maintained in future upgrades. By default the CDB$ROOT and PDB$SEED databases are always processed first. They are processed first even if they are not added to a priority list. All PDBs in the priority list are processed before PDBs not in the priority list.
`-M`	Keeps CDB$ROOT in UPGRADE mode while the PDBs are upgraded. For non-CDBs, this parameter is ignored. During CDB upgrades, using this parameter setting places the CDB and all its PDBs in upgrade mode, which can reduce total upgrade time. However, you cannot bring up any of the PDBs until the CDB and all its PDBs are upgraded. By default, if you do not use the `-M` parameter setting, then CDB$ROOT is upgraded and restarted in normal mode, and the normal background processes are started. As each PDB is upgraded, you can bring the PDB online while other PDBs are still being upgraded.
`-n`	Specifies the number of processes to use for parallel operations. Non-CDBs: The `-n` parameter specifies the number of SQL processes to use when upgrading the database. Multitenant architecture databases (CDBs): The number of PDBs upgraded concurrently is controlled by the value of the `-n` parameter. Multiple PDB upgrades are processed together. Starting in Oracle Database 12*c*, the default value for multitenant architecture databases is the number of CPUs on your system. A `cpu_count` equal to 24 equates to a default value of 24 for `-n`. Values for the `-n` parameter: Non-CDBs: The maximum value for `-n` is 8. The minimum value is 1. The default value is 4. Multitenant architecture databases (CDBs): The maximum value for `-n` is unlimited. The minimum value is 4. The maximum PDB upgrades running concurrently is the value of `-n` divided by the value of `-N`.
`-N`	Specifies the number of SQL processors to use when upgrading PDBs. For non-CDBs, this parameter is ignored. For CDBs, the maximum value is 8. The minimum value is 1. The default value is 2.

Table 4-1 (Cont.) Parallel Upgrade Utility (catctl.pl) Parameters

Parameter	Description
`-p`	Restarts from the specified phase. When you re-run an upgrade, it does not restart phases already completed successfully.
`-P`	Stops from the specified phase.
`-R`	Resumes the upgrade from a failed phase. Using the `-R` parameter enables the upgrade to resume at the point of failure, so that only the missing upgrade phases are rerun.
`-s`	Names the SQL script that initializes sessions.
`-S`	Specifies serial upgrade instead of parallel. Starting with Oracle Database 12.2, `catupgrd.sql` is no longer supported using the `-S` option.
`-T`	Takes offline user schema-based table spaces.
`-u`	Specifies user name, and prompts for password.
`-y`	Displays phases only.
`-z`	Turns on production debugging information for `catcon.pm`.
`-Z`	Turns on debug tracing information for `catctl.pl`. For example, to set the number to 1, enter `-Z 1`.

Example of Using the Parallel Upgrade Utility

Use this example to understand how you can run the parallel upgrade utility manually to perform upgrades.

The Parallel Upgrade Utility (`catctl.pl`) is integrated with AutoUpgrade and DBUA. The `catctl.pl` Perl script uses classic upgrade to upgrade `CDB$ROOT`. You can also run the Parallel Upgrade Utility manually by using the command-line script `dbupgrade`. Run the Parallel Upgrade Utility using the command-line parameters to specify how you want the upgrade to run. For example, to run the utility in serial mode instead of using parallel operations, specify the `-n 1` option.

Example 4-6 Running Parallel Upgrade Utility with Parameters for CDB and Non-CDB Databases

If you use the option `-n 4` when you run the Parallel Upgrade Utility, then the upgrade process creates `catupgrd0.log`, `catupgrd1.log`, `catupgrd2.log`, and `catupgrd3.log`. Check all of the `catupgrd#.log` files to confirm that the upgrade succeeded. If the upgrade failed, and you fix issues and run the Parallel Upgrade Utility again, then the previous log files are overwritten, unless you specify a different log directory by using the `-l` parameter.

For example:

```
cd $ORACLE_HOME/bin
dbupgrade -n 4 -l $ORACLE_HOME/diagnostics
```

Example 4-7 Running Parallel Upgrades on Multiple Pluggable Databases (PDBs) Using Parallel Upgrade Utility

These examples show how parameter settings change the way that the Parallel Upgrade Utility performs the upgrade on multiple PDBs.

> **Note:**
>
> In your upgrade plans, be aware of the following:
>
> - The `CDB$ROOT` defaults to a minimum value of 4 SQL processes, and to a maximum value of 8
> - The default value for `-N` is 2.
> - `PDB$SEED` always counts as one (1) PDB in the upgrade cycles
> - The default for the Parallel Upgrade Utility parameter `-n` is the value of the `CPU_COUNT` parameter

In the following examples, the system is an Oracle Multitenant Oracle Database system that has a `CPU_COUNT` value of 24.

Run the Parallel Upgrade Utility without specifying values for the parameters `-n` and `-N` (that is, accept the default value of -N, which is 2, and accept the default value of `-n` as the `CPU_COUNT` parameter value, which is 24). The following parallel processing occurs:

- 12 PDBs are upgraded in parallel (`CPU_COUNT` divided by 2)
- 2 parallel processes run for each PDB

Specify the value of `-n` as 64, and `-N` as 4. The following parallel processing occurs:

- 16 PDBs are upgraded together (64 divided by 4)
- 4 parallel processes run for each PDB

Specify the value of `-n` as 20, and `-N` as 2. The following parallel processing occurs:

- 10 PDBs are upgraded together (20 divided by 2)
- 2 parallel processes run for each PDB

Specify the value of `-n` as 10, and `-N` as 4. The following parallel processing occurs:

- 2 PDBs are upgraded together (10 divided by 4), rounded down.
- 4 parallel processes run for each PDB

Do not specify the value of `-n` (that is, accept the default value of `-n`, which is the value of the `CPU_COUNT` parameter), and specify the value of `-N` as 1. The following parallel processing occurs:

- 24 PDBs are upgraded together (`CPU_COUNT` value divided by 1)
- 1 process runs for each PDB

Specify a value for `-n` as 20, and do not specify the value for `-N` (that is, accept the default value of `-N`, which is 2). The following parallel processing occurs:

- 10 PDBs are upgraded together (20 divided by 2)

- 2 parallel processes run for each PDB

Manual Upgrade Scenarios for Multitenant Architecture Oracle Databases

To prepare for manual upgrades, review the manual upgrade scenarios and procedures for Oracle Database deployed with multitenant architecture.

Starting with Oracle Database 12c, multitenant architecture enables Oracle Database deployments using multitenant container databases (CDB) that contain pluggable databases (PDBs). All Oracle Database releases earlier than Oracle Database 12c Release 1 (12.1.0.1) use non-CDB architecture.

> **Caution:**
>
> You cannot downgrade a database after you have set the compatible initialization parameter to 12.1.0.2 or later releases. A downgrade is possible for a pluggable database (PDB) only if the compatibility is set to 12.1.0.1. There can still be restrictions on downgrading.
>
> Before starting an upgrade, and before starting a downgrade, Oracle strongly recommends that you upgrade your source and target databases to the most recent release update or release update revision.

- About Oracle Multitenant Oracle Database Upgrades
  You can upgrade Oracle Databases installed on multitenant architecture either in parallel, or in sequence.
- Coordinate Upgrades of Proxy PDBs with Multitenant Upgrades
  Coordinate upgrades of the CDB so that proxy PDB and PDB targets are the same version.
- Manually Upgrading a Multitenant Container Oracle Database (CDB)
  The procedure in this section provides steps for upgrading a CDB manually using a command-line procedure.
- About Upgrading PDBs Using the Parallel Upgrade Utility with Priority Lists
  In Oracle Database 12.2 and later releases, you can upgrade PDBs using a priority list to upgrade a set of PDBs ahead of other PDBs, and you can modify that upgrade priority.
- About PDB Upgrades Using Priority Lists, Inclusion Lists, and Exclusion Lists
  To control how your pluggable databases (PDBs) are upgraded, you can use inclusion and exclusion lists with priority lists.
- Upgrading Multitenant Architecture In Parallel
  Use this technique to upgrade multitenant architecture Oracle Database releases (Oracle Database 12c Release 1 (12.1.0.1) and later by upgrading container databases (CDBs), and then upgrading multiple pluggable databases (PDBs) in parallel.
- Upgrading Multitenant Architecture Sequentially Using Unplug-Plug
  To upgrade pluggable databases (PDBs) that are in an earlier release multitenant container databases (CDBs), Oracle Database Release 12*c* (12.1.0.1) and later,

you can unplug the PDBs from the earlier release CDB, and plug the PDBs into the later release CDB.

Related Topics

- *Oracle Multitenant Administrator's Guide*

About Oracle Multitenant Oracle Database Upgrades

You can upgrade Oracle Databases installed on multitenant architecture either in parallel, or in sequence.

Starting with Oracle Database 12c release 1 (12.1), Oracle provided multitenant architecture, which enables the creation and administration of pluggable databases (PDBs) in a container database (CDB). You can upgrade multitenant architecture systems using either Oracle Database Upgrade Assistant (DBUA), or using the Parallel Upgrade Utility to carry out manual upgrades.

There are two techniques for upgrading Oracle Databases using the multitenant architecture:

- **In parallel**. With this technique, you carry out one upgrade operation that upgrades the CDB, and then upgrades the PDBs in parallel.
- **Sequentially**. With this technique, you install a new release CDB, prepare and unplug PDBs from the earlier release CDB, plug the PDBs into a later release CDB, and then complete the upgrade for each PDB.

The following sections provide a high-level summary of each upgrade technique.

Upgrading Oracle Multitenant In Parallel

With the In Parallel technique, you first upgrade `CDB$ROOT` using the Parallel Upgrade Utility (`catctl.pl`), using parameters to set the degree of parallel processing and availability:

- The `-n` parameter defines how many parallel processes run the upgrade, up to 8.
- The `-M` parameter determines if the `CDB$ROOT` stays in `UPGRADE` mode through the entire upgrade, or becomes available for access after the CDB upgrade is complete. If you do not run the upgrade with the `-M` parameter, then when the `CDB$ROOT` upgrade is complete, PDBs then become available for access as soon as each PDB completes its upgrade. If you run the upgrade with the `–M` parameter, then `CDB$ROOT` stays in `UPGRADE` mode, and PDBs do not become available until upgrade of all PDBs is complete.

Upgrading Oracle Multitenant In Sequence

With the In Sequence technique, you install the new release multitenant architecture CDB. Next, in the earlier release multitenant architecture CDB, you issue SQL commands to run preupgrade scripts to prepare one or more PDBs to upgrade, and shut them down. You then unplug PDBs, plug them into the new release multenant architecture CDB, and complete the upgrade sequentially for each PDB.

Coordinate Upgrades of Proxy PDBs with Multitenant Upgrades

Coordinate upgrades of the CDB so that proxy PDB and PDB targets are the same version.

During upgrades, upgrade of a Proxy PDB does not upgrade its corresponding target PDB. Upgrade of the target PDB has to be done separately.

Manually Upgrading a Multitenant Container Oracle Database (CDB)

The procedure in this section provides steps for upgrading a CDB manually using a command-line procedure.

You must complete the following steps before using this procedure:

- Install the new release software for Oracle Database
- Prepare the new Oracle home
- Run the Pre-Upgrade Information Tool

Oracle Database 12c introduced multitenant architecture, which enables Oracle Database to function as a multitenant container database (CDB) with pluggable databases. You can upgrade the CDB using DBUA, and in the process, upgrade all the pluggable databases attached to the CDB at the same time. Or, after you install the software for Oracle Database and prepare the new Oracle home, you can proceed with a manual, command-line upgrade.

1. If you have not done so, run the Pre-Upgrade Information Tool. Review the Pre-Upgrade Information tool output and correct all issues noted in the output before proceeding.
2. Back up the source database.
3. If you have not done so, prepare the new Oracle home.
4. (Conditional) For Oracle RAC environments only, enter the following commands to set the initialization parameter value for CLUSTER_DATABASE to FALSE:

   ```
   ALTER SYSTEM SET CLUSTER_DATABASE=FALSE SCOPE=SPFILE;
   ```

   Restart the database after changing the `CLUSTER_DATABASE` parameter.
5. Shut down the database. (The syntax is the same for a non-CDB and a CDB.)

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

   > **Note:**
   >
   > To close a PDB, you can specify it from the CDB root: `alter pluggable database` *`PDBname`* `close`.

6. If your operating system is Windows, then complete the following steps:
   a. Stop the `OracleService` *`SID`* 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`.

If your *SID* is ORCL, then enter the following command, substituting your SID for *SID*.

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

c. Create the service for the new release Oracle Database at a command prompt using the ORADIM command of the new Oracle Database release.

For example:

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

Most Oracle Database services log on to the system using the privileges of the Oracle Home User. The service runs with the privileges of this user. The ORADIM command prompts you for the password to this user account. You can specify other options using ORADIM.

In this example, if your *SID* value is ORCL, your *password* (SYSPWD) value is TWxy5791, the maximum number of users (MAXUSERS) value is 10, and the Oracle home path is C:\ORACLE\PRODUCT\19.1.0\DB, then enter the following command:

```
C:\> ORADIM -NEW -SID ORCL -SYSPWD TWxy5791 -MAXUSERS 10
-STARTMODE AUTO -PFILE
C:\ORACLE\PRODUCT\19.1.0\DB\DATABASE\INITORCL.ORA
```

ORADIM writes a log file to the ORACLE_HOME\database directory. The log file contains the name of the PDB in the multitenant database.

7. If your operating system is Linux or UNIX, then perform the following checks:
   a. Your ORACLE_SID is set correctly
   b. The oratab file points to the Oracle home for Oracle Database 12*c*
   c. The following environment variables point to the Oracle Database 12c directories:
      - ORACLE_HOME
      - PATH
   d. Any scripts that clients use to set $ORACLE_HOME environment variable must point to the new Oracle home.

> **Note:**
>
> If you are upgrading an Oracle Real Application Clusters database, then perform these checks on all nodes where the Oracle Real Application Clusters database has instances configured.

> **See Also:**
>
> Oracle Database and Oracle Clusterware installation guides for information about setting other important environment variables on your operating system

8. Log in to the system as the owner of the Oracle home under the new Oracle Database release.
9. Start SQL*Plus in the new Oracle home from the path *Oracle_home*/rdbms/admin directory.

   For example:

   ```
   $ cd $ORACLE_HOME/rdbms/admin
   $ pwd
   /u01/app/oracle/product/19.0.0/dbhome_1/rdbms/admin
   $ sqlplus
   ```

   On Windows platforms, to access SQL*Plus, change directory to `%ORACLE_HOME%/bin`
10. Connect to the database that you want to upgrade using an account with SYSDBA privileges:

    ```
    SQL> CONNECT / AS SYSDBA
    ```

11. Start the CDB in upgrade mode:

    ```
    SQL> startup upgrade
    ```

12. Start the instance by issuing the following command in SQL*Plus:

    ```
    SQL> alter pluggable database all open upgrade;
    ```

    If errors appear listing desupported initialization parameters, then make a note of the desupported initialization parameters and continue with the upgrade. Remove the desupported initialization parameters the next time you shut down the database.

> **Note:**
>
> Starting up the database in `UPGRADE` mode does the following:
>
> - Starts up the database with a new version of the Oracle Database instance
> - Restricts logins to SYSDBA
> - Disables system triggers
> - Performs additional operations that prepare the database for upgrades

13. Exit SQL*Plus before proceeding to the next step.

    For example:

    ```
    SQL> EXIT
    ```

14. To upgrade an entire CDB, run the Parallel Upgrade Utility (`catctl.pl`) from the new Oracle home. The Parallel Upgrade Utility provides parallel upgrade options that reduce downtime. You can run the command by using the command-line script `dbupgrade` from the new Oracle home.

    For example:

    Linux:

    ```
    cd $ORACLE_HOME/bin
    ./dbupgrade
    ```

    Windows:

    ```
    cd %ORACLE_HOME%\bin
    dbupgrade
    ```

    > **Note:**
    >
    > - Use the `-d` option to specify the directory that contains the files that you want to process. Use the `-l` option to specify the directory that you want to use for spool log files.
    > - If you are upgrading an entire CDB, and there are errors in `CDB$ROOT`, then the upgrade aborts.

15. To upgrade a subset of PDBs within a CDB, specify either an inclusion list, or an exclusion list.

    - This example for a Linux or UNIX system uses an inclusion list to upgrade `PDB1` only:

      ```
      cd $ORACLE_HOME/bin
      ./dbupgrade -c 'PDB1'
      ```

    - This example for a Windows system uses an exclusion list to upgrade everything in the CDB except `PDB1`:

      ```
      cd $ORACLE_HOME\bin
      dbupgrade -C "PDB1"
      ```

> **Note:**
>
> You can upgrade an individual PDB by unplugging it from the earlier release CDB, and plugging it into a later release CDB.
>
> For Windows, when you run the `dbupgrade` command with the inclusion (`-c`) or the exclusion (`-C`) options, you must specify the option with quotes around the CDB root name and PDB seed name.
>
> For example:
>
> `... -C "CDB$ROOT PDB$SEED"`

16. For CDBs, log in to the CDB as SYSDBA and run the command alter pluggable database all open to make databases available for recompiling code. For example:

```
$ sqlplus / as sysdba

SQL> alter pluggable database all open;
```

17. Run `catcon.pl`. This command starts `utlrp.sql` and recompiles any remaining stored PL/SQL and Java code.

    For example:

```
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlrp -d '''.''' 
utlrp.sql
```

    Because you run the command using `-b utlrp0`, the log file `utlrp0.log` is generated with the recompile results.

18. Run `postupgrade_fixups.sql`.

    For example:

```
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b postupgrade_fixups
-d '''.''' postupgrade_fixups.sql
```

19. Run `utlusts.sql`. This command verifies that all issues are fixed.

    For example, in a CDB:

```
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlu122s -d '''.''' 
utlusts.sql
```

    Because you run the command using `-b utlu122s`, the log file `utlu122s0.log` is generated with the upgrade results. You can review the `upg_summary.log` file to review the upgrade report.

    To see information about the state of the database, run `utlusts.sql` as many times as you want, at any time after you complete the upgrade. If the `utlusts.sql` script returns errors, or if it shows components that are not marked as `VALID`, or if the SQL script you run is not from the most recent release, then refer to the troubleshooting section in this guide.

20. Ensure that the time zone data files are current by using the `DBMS_DST` PL/SQL package to upgrade the time zone file. You can also update the time zone after the upgrade. If you update the time zone, then you must update the time zone in both `CDB$ROOT` and the PDBs.
21. Exit SQL*Plus.

    For example:

    ```
    SQL> EXIT
    ```

22. (Conditional) If you are upgrading an Oracle Real Application Clusters database, then use the following command syntax to upgrade the database configuration in Oracle Clusterware:

    ```
    $ srvctl upgrade database -db db-unique-name -oraclehome oraclehome
    ```

    In this example, *`db-unique-name`* is the assigned database name (not the instance name), and *`oraclehome`* is the Oracle home location in which the database is being upgraded. The `SRVCTL` utility supports long GNU-style options, in addition to the short CLI options used in earlier releases.

Your database is now upgraded. You are ready to complete post-upgrade procedures.

Caution:

If you retain the old Oracle software, then never start the upgraded database with the old software. Only start Oracle Database using the start command in the new Oracle Database home.

Before you remove the old Oracle environment, relocate any data files in that environment to the new Oracle Database environment.

See Also:

Oracle Database Administrator's Guide for information about relocating data files

Related Topics

- *Oracle Database Administrator's Guide*

About Upgrading PDBs Using the Parallel Upgrade Utility with Priority Lists

In Oracle Database 12.2 and later releases, you can upgrade PDBs using a priority list to upgrade a set of PDBs ahead of other PDBs, and you can modify that upgrade priority.

Priority lists enable you to group and upgrade PDBs according to their priority. A priority list is a text file with comma-delimited lists defining the order of upgrade priority, and the PDBs in each numeric priority group. You run the Parallel Upgrade Utility (`dbupgrade`, `dbupgrade.cmd`, or `catctl.pl`) using the `-L` option to run the upgrade using a priority list, and to call that list as the upgrade runs.

Create the list using the following format. In this format example, the variable *numeral* is a numeric value, and *pdbx* is the name of a PDB.

Number, Pdb
numeral,*pdb1*,*pdb2*,*pdb3*
numeral,*pdb4*
numeral,*pdb5*,*pdb6*,*pdb7*,*pdb8*
.
.
.

The numeral represents the priority for the PDB.

PDB priorities are as follows:

1. CDB$ROOT: Priority 1. Upgrading the container database first is a mandatory priority. You cannot change the priority for the container database upgrade. CDB$ROOT is always processed first.
2. PDB$SEED: Priority 1. Upgrading the PDB seed database is a mandatory priority. You cannot change the priority for the PDB seed upgrade. PDB$SEED always upgraded after CDB$ROOT, and with the first batch of PDB upgrades.
3. Priority List 1 PDBs: Priority 1 is the highest user-selected priority. These PDBs are upgraded second after CDB$ROOT, in the batch where the PDB$SEED PDB is upgraded.
4. Priority List 2 PDBs: Priority 2 is the second-highest priority PDB set. These PDBs are upgraded after the Priority 1 PDBs.
5. Priority List 3 PDBs: Priority 3 is the third-highest priority PDB set. These PDBS are upgraded after priority 2 PDBs.
6. Priority List 4 PDBs: Priority 4 is the fourth-highest priority PDB set. These PDBS are upgraded after priority 3 PDBs.
7. Priority List 5 PDBs: Priority 5 is the fifth-highest priority PDB set. These PDBS are upgraded after priority 4 PDBs.
8. Priority List 6 PDBs: Priority 6 is the sixth-highest priority PDB set. These PDBS are upgraded after priority 5 PDBs.

When you run the Parallel Upgrade Utility, the following processing rules apply:

- CDB$ROOT and PDB$SEED are always processed first, even if they are not present in the priority list.

- All PDBs that are in priority lists are processed in order of priority
- Any PDBs that are not listed in priority lists are processed after the PDBs named in the priority list.

For example:

```
Number,Pdb
1,sales1,region2,receivables1
2,sales2
3,dss1,region3,region2,dss2,dss3
```

Use the following syntax to run the Parallel Upgrade utility using a priority list:

```
dbupgrade -L priority_list_name
```

For example, to run the Parallel Upgrade Utility on a Windows system using the Parallel Upgrade Utility batch command and a priority list named `My122Upgrade`, enter the following command:

```
C:>\u01\app\19.1.0\db_home1\rdbms\admin\dbupgrade -L MyUpgrade
```

After you complete an upgrade using a priority list to set upgrade priorities, these PDB priority states are maintained in the CDB for the PDBs. The next upgrade honors the priorities set for the PDBs in the previous upgrade.

Use the following SQL command syntax to change PDB upgrade priority states, where *`PDBName`* is the name of the PDB whose upgrade priority you want to change, and *`PDBPriorityNumber`* is the new priority value you want to assign:

```
SQL> alter session set container = CDB$ROOT
SQL> alter pluggable database PDBName upgrade priorityPDBPriorityNumber
```

For example:

```
SQL> alter session set container = CDB$ROOT
SQL> alter pluggable database region2 upgrade priority 2
```

In this example, the PDB named region 2 that was set to upgrade priority 1 in the previous example is changed to upgrade priority 2.

About PDB Upgrades Using Priority Lists, Inclusion Lists, and Exclusion Lists

To control how your pluggable databases (PDBs) are upgraded, you can use inclusion and exclusion lists with priority lists.

With Oracle Database 12c Release 1 (12.1.0.2) and later releases, the preferred method for specifying the order in which upgrades are applied to PDBs is to set the priority on the source Oracle Database using `alter pluggable database` with `upgrade`

`priority`, where *`pdbname`* is the PDB, and *`number`* is the priority that you want to assign for upgrade:

```
alter pluggable database pdbname upgrade priority number
```

For example:

```
alter pluggable database CDB1_PDB3 upgrade priority 2
```

However, you can also specify priority lists at the time of upgrade by using the procedures described here.

Upgrade Processing and Lists

The following terms designate types of upgrade list processing:

- **Priority lists**: Comma-delimited lists that designate the upgrade priority of PDBs in the list.
- **Inclusion lists**: Comma-delimited lists that designate PDBs that you want to upgrade. PDBs in these lists are upgraded after the PDBs listed in priority lists.
- **Exclusion lists**: Comma-delimited lists that designate PDBs that you do not want to be upgraded.

You can use inclusion lists and exclusion lists in the following ways:

- On their own, to include or exclude a set of PDBs from an upgrade
- In conjunction with priority lists to provide detailed specifications for the order in which PDBs are upgraded, and which PDBs are excluded from an upgrade.

  When inclusion lists are used with priority lists, the PDBs listed in inclusion lists are upgraded according to the priority value they are assigned in the priority lists. PDBs listed in inclusion lists but not listed in priority lists are upgraded after all PDBs in the priority lists are upgraded.

  When exclusion lists are used with priority lists, the PDBs listed in exclusion lists are not upgraded.

> **Note:**
>
> Create priority lists using a plain text editor, such as `vi` on Linux and Unix, or Notepad on Microsoft Windows.
>
> In the examples in this topic, the `cpu_count` value is equal to 2.

Upgrade Priority using Default Processing

Default processing is the upgrade processing that is carried out if you do not designate how you want to upgrade PDBs in your container databases (CDBs) using lists.

With default processing, `CDB$ROOT` is upgraded, and then `PDB$SEED`. Depending on the degree of parallelism you set, one or more PDBs may be updated in parallel

with PDB$SEED. As upgrades complete, PDBs are upgraded as upgrade processors become available.

The examples that follow use the following multitenant configuration of CDB and PDBs:

```
CDB$ROOT
PDB$SEED
CDB1_PDB1
CDB1_PDB2
CDB1_PDB3
CDB1_PDB4
CDB1_PDB5
```

In default processing, you specify no preference for which PDBs you want upgraded or excluded from upgrade. With default processing, CDB$ROOT is upgraded first, and PDB$SEED is updated in the first group of PDBs upgraded.

Example 4-8 Specifying Complete PDB Upgrade Priority

The following example of a priority list, where the priority setting for all PDBs is set by the list:

```
1,CDB$ROOT
1,PDB$SEED
1,CDB1_PDB1
1,CDB1_PDB2
2,CDB1_PDB3
2,CDB1_PDB4
3,CDB1_PDB5
```

Here is another way of writing the same list, in which you group PDBs in priority order:

```
1,CDB$ROOT
1,PDB$SEED
1,CDB1_PDB1,CDB1_PDB2
2,CDB1_PDB3,CDB1_PDB4
3,CDB1_PDB5
```

In the preceding example, the PDBs listed in priority 1 are CDB1_PDB1 and CDB1_PDB2. These PDBs are upgraded before CDB1_PDB3 and CDB1_PDB4.

Here is another way of writing the same list, using container ID values (CON_ID) to set the priority order:

```
1,CDB$ROOT
1,PDB$SEED
1,3,4
2,5,6
3,7
```

In the preceding example, the PDBs listed in priority 1 are CDB1_PDB1 (identified by CON_ID 3) and CDB1_PDB2 (identified by CON_ID 4). These PDBs are upgraded before CDB1_PDB3 (CON_ID 5) and CDB1_PDB4 (CON_ID 6).

When you use the CON_ID method to specify priority, the first number specifies the priority of the group of PDBs. The second value or number specifies the PDBs (by CON_ID) number that are in that priority grouping. CDB$ROOT is always updated first, and PDB$SEED is always updated in the first upgrade priority group.

These examples all show a priority list upgrade with the following characteristics:

- Exclusion processing: None
- Inclusion processing: None
- Default processing: None

The upgrade order is carried out in the following sequence:

1. CDB$ROOT
2. PDB$SEED, CDB1_PDB1
3. CDB1_PDB2, CDB1_PDB3
4. CDB1_PDB4, CDB1_PDB5

Example 4-9 Specifying a Priority Subset of PDBs, and Upgrading Other PDBs with Default Processing

The following example specifies a priority list called priority.lst, which specifies a subset of PDBs for upgrade:

```
catctl -L priority.lst catupgrd.sql
1,CDB$ROOT
1,PDB$SEED
1,CDB1_PDB1,CDB1_PDB2
```

This example shows a priority list upgrade with the following characteristics:

- Exclusion processing: None
- Inclusion processing: None
- Default processing: CDB1_PDB3, CDB1_PDB4, CDB1_PDB5

The upgrade order is carried out in the following sequence:

1. CDB$ROOT
2. PDB$SEED, CDB1_PDB1
3. CDB1_PDB2, CDB1_PDB3
4. CDB1_PDB4, CDB1_PDB5

Example 4-10 Specifying a Priority Subset of PDBs, and Upgrading Other PDBs with an Inclusion List

The following example specifies a priority list called priority.lst, which specifies a priority subset of PDBs for upgrade:

```
catctl -L priority.lst -c 'CDB1_PDB2 CDB1_PDB4 CDB1_PDB5' catupgrd.sql
```

This command refers to the following priority list:

```
1,CDB$ROOT
1,PDB$SEED
1,CDB1_PDB2,CDB1_PDB4
2.CDB1_PDB5
```

This example shows a priority list upgrade with the following characteristics:

- Exclusion processing: None
- Inclusion processing: `CDB1_PDB2`, `CDB1_PDB4`, `CDB1_PDB5`
- Default processing: None

The upgrade order is carried out in the following sequence:

1. `CDB1_PDB2`, `CDB1_PDB4`
2. `CDB1_PDB5`

The Parallel Upgrade Utility processes only the PDBs that are in the inclusion list, and in the order of the priority list.

Example 4-11 Specifying a Priority Subset of PDBs, and Excluding CDB$ROOT with an Exclusion List

The following example runs `catctl` using a priority list called `priority.lst`. Because this command runs with the `-C` option, it excludes CDB$ROOT from the upgrade:

```
catctl -L priority.lst -C 'CDB$ROOT' catupgrd.sql
```

This is the priority list:

```
1,CDB$ROOT
1,PDB$SEED
1,CDB1_PDB1,CDB1_PDB2
2,CDB1_PDB3,CDB1_PDB4
3,CDB1_PDB5
```

The upgrades are processed using the priority list to specify upgrade priority.

- Inclusion processing: None
- Exclusion processing: `CDB$ROOT`
- Priority processing: `PDB$SEED`, `CDB1_PDB1`, `CDB1_PDB2`, `CDB1_PDB3`, `CDB1_PDB4`, `CDB1_PDB5`

Because `CDB$ROOT` is excluded, the priority processing shifts. The upgrade order is carried out in the following sequence:

1. `PDB$SEED`, `CDB_PDB1`
2. `CDB_PDB2`, `CDB_PDB3`
3. `CDB1_PDB4`, `CDB1_PDB5`

Example 4-12 Specifying an Exclusion List using CATCTL_LISTONLY

The following example specifies a priority list called `priority.lst`, which specifies a subset of PDBs for upgrade. With the `CATCTL_LISTONLY` option, PDBs that are not in the priority list are excluded from the upgrade:

```
catctl -L priority.lst -C 'CATCTL_LISTONLY' catupgrd.sql
```

Priority list:

```
1,CDB$ROOT
1,PDB$SEED
1,CDB1_PDB1,CDB1_PDB2
2,CDB1_PDB3
3,CDB1_PDB5
```

- Exclusion processing: `CATCTL_LISTONLY` (Only process inclusion priority list)
- Inclusion processing: None
- Default processing: None

The upgrade order is carried out in the following sequence:

1. `CDB$ROOT`
2. `PDB$SEED`, `CDB1_PDB1`, `CDB1_PDB2`
3. `CDB1_PDB3`, `CDB1_PDB5`

> **Note:**
>
> Specifying the keyword `CATCTL_LISTONLY` in the exclusion list turns the priority list into an inclusion priority list. Only PDBs in the list are processed. No default processing occurs in this scenario, so in this example, `CDB1_PDB4` is not processed.

Example 4-13 Specifying a Priority List using CON_ID Values

The following example specifies a priority list called `priority.lst`, which specifies a subset of PDBs for upgrade:

```
catctl -L priority.lst -C 'CATCTL_LISTONLY' catupgrd.sql
```

The upgrade order is determined by the priority list priority number. In the list called by the `-L` parameter, `priority.lst`, the numbers following the upgrade priority number are the CON_ID values associated with PDB names:

```
1,3,4
2,5,CDB1_PDB4
3,7
```

In the preceding list example, note that you can use a mix of CON_ID numbers and PDB names.

The PDBs listed in priority 1 are `CDB1_PDB1` (identified by `CON_ID 3`) and `CDB1_PDB2` (identified by `CON_ID 4`). These PDBs are upgraded before `CDB1_PDB3` (`CON_ID 5`), `CDB1_PDB4`, which is identified by name, and `CDB1_PDB5` (`CON_ID 7`).

- Exclusion processing: `-C CATCTL_LISTONLY` (Only process PDBs in the inclusion priority list)
- Exclusion Processing: None
- Inclusion processing: Specified in `priority.lst`
- Default processing: `CDB$ROOT`, `PDB$SEED`

The upgrade order is determined by the priority list, which uses the `CON_ID` numbers associated with the PDB.

1. `CDB$ROOT`
2. `PDB$SEED`, `CDB1_PDB1`
3. `CDB1_PDB2`, `CDB1_PDB3`
4. `CDB1_PDB4`, `CDB1_PDB5`

> **Note:**
>
> This example demonstrates the use of the `CON_ID` method to specify the PDBs, and omits `CDB$ROOT` and `PDB$SEED` from the priority list. `CDB$ROOT` and `PDB$SEED` are processed using default processing.

Upgrading Multitenant Architecture In Parallel

Use this technique to upgrade multitenant architecture Oracle Database releases (Oracle Database 12c Release 1 (12.1.0.1) and later by upgrading container databases (CDBs), and then upgrading multiple pluggable databases (PDBs) in parallel.

- About Upgrading Pluggable Databases (PDBs) In Parallel
  Using the In-Parallel technique, you can upgrade the container database (CDB), and then immediately upgrade PDBs using parallel SQL processors.
- Upgrading Multitenant Container Databases In Parallel
  Use this technique to upgrade `CDB$ROOT`, `PDB$SEED`, and all PDBs in the CDB in one upgrade operation.

About Upgrading Pluggable Databases (PDBs) In Parallel

Using the In-Parallel technique, you can upgrade the container database (CDB), and then immediately upgrade PDBs using parallel SQL processors.

CDBs can contain zero, one, or more PDBs. By default, the Parallel Upgrade Utility (`catctl.pl`) updates the CDB and all of its PDBs in the same upgrade window. The Parallel Upgrade Utility uses the number of computer processing units (CPUs) to determine the maximum number of PDBs that are upgraded simultaneously. The

number of PDBs that are upgraded in parallel is determined by dividing the parallel SQL process count (`-n` option) by the parallel PDB SQL process count (`-N` option).

> **Note:**
>
> You must plan your upgrade window to accommodate a common downtime for all of the database services that the PDBs on the CDB are providing.

Pluggable Database Upgrade Syntax

```
dbupgrade [-M] -n [-N]
```

- `-M` Specifies if `CDB$ROOT` is kept in upgrade mode, or if it becomes available when it completes upgrade:
  - If you run the Parallel Upgrade Utility with the `-M` parameter, then the upgrade places `CDB$ROOT` and all of its PDBs in upgrade mode, which can reduce total upgrade time. However, you cannot bring up any of the PDBs until the CDB and all of its PDBs are upgraded.
  - If you do not run the Parallel Upgrade Utility with the `-M` parameter, then CDB$ROOT is upgraded and restarted in normal mode, and the normal background processes are started. After a successful upgrade, only `CDB$ROOT` is opened in read/write mode. All the PDBs remain in `MOUNT` mode. As each PDB is upgraded, you can bring each PDB online while other PDBs are still being upgraded.
- `-n` Specifies the number of in-parallel PDB upgrade processors.

  If you do not specify a value for `-n`, then the default for `-n` is the `CPU_COUNT` value.

  If you do specify a value for `-n`, then that value is used to determine the number of parallel SQL processes. The maximum value is unlimited. The minimum value is 4.
- `-N` Specifies the number of SQL processors to use when upgrading PDBs. The maximum value is 8. The minimum value is 1. If you do not specify a value for `-N`, then the default value is 2.
- The maximum PDB upgrades running concurrently is the value of `-n` divided by the value of `-N`.

The following is a high-level list of actions during the In Parallel PDB upgrade technique:

1. Make sure that your backup strategy is complete.
2. Run the Pre-Upgrade tool. Fix any issue that is reported.. The Pre-Upgrade Tool (`preupgrade.jar`) is shipped with the new Oracle database release.
3. Run the Parallel Upgrade Utility. In sequence, the following upgrades are carried out:
   a. Cycle 1: `CDB$ROOT` is upgraded to the new Oracle release
   b. Cycle 2 to Cycle *x*: `PDB$SEED` and PDBs are upgraded in parallel, with the number of cycles of upgrades as determined by the parameter settings you specify with `-n`.
4. Complete post-upgrade steps.

Example 4-14 Example of Multitenant Architecture Upgrade Using Defaults (No Parameters Set)

In this scenario, your `CPU_COUNT` value is equal to 24. If you do not specify a value for in-parallel PDB processors using the `-n` option, then the default value for in-parallel PDB processors (`-n`) is equal to 24. If you do not specify a value for `-N`, then the default value for the number of SQL processors (`-N`) is 2.

Result:

12 PDBs are upgraded in parallel (`CPU_COUNT` divided by 2, or 24 divided by 2.) There are 2 parallel SQL processes allocated for each PDB.

Example 4-15 Example of Multitenant Architecture Upgrade Using 64 In Parallel PDB Upgrade Processors and 4 Parallel SQL Processes

In this scenario you set the value of in-parallel PDB upgrade processors to 64 by specifying the option `-n 64`. You specify the value of parallel SQL processors to 4 by specifying the option `-N 4`.

Result:

16 PDBs are upgraded in parallel (64 divided by 4). There are 4 parallel SQL processes for each PDB.

Example 4-16 Example of Multitenant Architecture Upgrade Using 20 In Parallel PDB Upgrade Processors and 2 Parallel SQL Processes

In this scenario you set the value of in-parallel PDB upgrade processors to 20 by specifying the option `-n 20`. You specify the value of parallel SQL processors to 2 by specifying the option `-N 2`.

Result:

10 PDBs are upgraded in parallel (20 divided by 2). There are 2 parallel SQL processes for each PDB.

Example 4-17 Example of Multitenant Architecture Upgrade Using 10 In Parallel PDB Upgrade Processors and 4 Parallel SQL Processes

In this scenario you set the value of in-parallel PDB upgrade processors to 10 by specifying the option `-n 10`. You specify the value of parallel SQL processors to 2 by specifying the option `-N 4`.

Result:

2 PDBs are upgraded in parallel (10 divided by 4). There are 4 parallel SQL processes for each PDB.

Upgrading Multitenant Container Databases In Parallel

Use this technique to upgrade `CDB$ROOT`, `PDB$SEED`, and all PDBs in the CDB in one upgrade operation.

If you do not choose to use the AutoUpgrade utility, then Oracle recommends that you use this approach if you can schedule downtime. Using this procedure upgrades in parallel all the PDBs in the multitenant architecture container database, depending on your server's available processors (CPUs). This is a direct procedure for upgrades that provides simplicity of maintenance.

> **Note:**
>
> When you upgrade the entire container using the In Parallel upgrade method, all the PDBs must be down. Perform the upgrade in a scheduled upgrade window so that you can bring all the PDBs down.

> **Caution:**
>
> - Always create a backup of existing databases before starting any configuration change.
> - You cannot downgrade a database after you have set the compatible initialization parameter.
> - Oracle strongly recommends that you upgrade your source and target databases to the most recent release update (RU) or release update revision (RUR) before starting an upgrade, and to the most recent release update before starting a downgrade.

1. Ensure that you have a proper backup strategy in place.
2. Open all PDBs.

   For example:

   ```
   SQL> alter pluggable database all open;
   ```

3. Run the Pre-Upgrade Information Tool (`preupgrade.jar`), using the following syntax:

   `/java -jar` *`$New_release_Oracle_home`*`/rdbms/admin/preupgrade.jar [TERMINAL|FILE|DIR outputdir] [TEXT|XML] [-c` *`InclusionListOfPDBs`*`] [-C` *`ExclusionListOfPDBs`*`]`
   Use space-delimitation for lists. On Linux and UNIX, define the list by placing the list inside single quotes: `'`. On Windows systems, define the list by placing the list inside double quotes `"`.

   For example, run the following command to run the Pre-Upgrade Information tool on PDBs PDB1 through PDB25, where you have set up an environment variable `$ORACLE_HOME_12.2` for your Oracle Database Oracle home in `/u01/app/oracle/product/12.2.0/dbhome_1/`, and you have set up an environment variable `$ORACLE_HOME_19.1` for your new Oracle Database Oracle home in `/u01/app/oracle/product/19.1.0/dbhome_1/`:
   Linux and UNIX:

   ```
   java -jar $ORACLE_HOME_19.1/rdbms/admin/preupgrade.jar \
   -c 'pdb1 pdb2 pdb3 pdb4 pdb5 pdb6 pdb7 pdb8 pdb9 pdb10 pdb11 pdb12
   pdb13\
   pdb14 pdb15 pdb16 pdb17 pdb18 pdb19 pdb20 pdb21 pdb22 pdb23 pdb24
   pdb25'
   ```

Windows:

```
java -jar %ORACLE_HOME_19.1%/rdbms/admin/preupgrade.jar \
-c "pdb1 pdb2 pdb3 pdb4 pdb5 pdb6 pdb7 pdb8 pdb9 pdb10 pdb11 pdb12
pdb13\
pdb14 pdb15 pdb16 pdb17 pdb18 pdb19 pdb20 pdb21 pdb22 pdb23 pdb24
pdb25"
```

> **Note:**
>
> You must use Java 1.5 or later to run the Pre-Upgrade Information tool. By default, the Java releases in Oracle Database releases that you can upgrade directly support the tool.

4. Review any generated fixup scripts and log files.

   By default, if ORACLE_BASE is defined, then the fixup files are placed in one of the following paths:

   - Linux and UNIX:

     `$ORACLE_BASE/cfgtoollogs/`*`db_unique_name`*`/preupgrade`

   - Windows

     `%ORACLE_BASE%\cfgtoollogs\`*`db_unique_name`*`\preupgrade`

   If ORACLE_BASE is not defined, then fixup files are placed in one of the following paths:

   - Linux and UNIX:

     `$ORACLE_HOME/cfgtoollogs/`*`db_unique_name`*`/preupgrade`

   - Windows:

     `%ORACLE_HOME\cfgtoollogs\`*`db_unique_name`*`\preupgrade`

   On multitenant architecture Oracle Databases, the Pre-Upgrade Information Tool also creates a consolidated `preupgrade_fixups.sql` script. You can run the consolidated fixup script by using `catcon.pl`. The consolidated fixup script runs on every container that was open at the time that you ran the `preupgrade.jar` command.

5. Run the `preupgrade_fixups` script, or individual PDB scripts. The `preupgrade_fixups` SQL scripts resolve some of the issues reported by the `preupgrade` script.

   On multitenant environment Oracle Database deployments, you can run `preupgrade_fixups`*`pdb-name`*`.sql` scripts on the source database, where *`pdb-name`* is the PDB name. If you generate fixup scripts for PDBs, then the PDB name is added to the fixup filename.

   In addition to the individual PDB fixup scripts, you can use `catcon.pl` to run the consolidated `preupgrade_fixups.sql` script. The consolidated script runs on every container that was open at the time that you ran `preupgrade.jar`.

> **Note:**
>
> Because $ is a reserved symbol on operating systems, the fixup script for PDB$SEED is `preupgrade_fixups_pdb_seed.sql`.

Complete any other preupgrade tasks that the Pre-Upgrade Information Tool identifies.

6. (Conditional) for Oracle RAC databases, set the cluster database initialization parameter to false:

   For example;

   ```
   ALTER SYSTEM SET cluster_database=FALSE SCOPE=spfile;
   ```

7. Shut down the database in the old Oracle home.

   For example, where *db_unique_name* is your database name:

   ```
   $ srvctl stop database -d db_unique_name
   ```

8. Copy the `PFILE` or `SPFILE` from the old Oracle home to the new Oracle home
9. Connect with SQL*Plus:

   ```
   sqlplus / as sysdba
   ```

10. Bring the CDB$ROOT instance into upgrade mode:

    ```
    STARTUP UPGRADE
    ```

11. Bring all PDBs into upgrade mode:

    ```
    ALTER PLUGGABLE DATABASE ALL OPEN UPGRADE;
    ```

12. Check the status of PDBs to confirm that they are ready to upgrade:

    ```
    SHOW PDBS
    ```

    For all PDBs, ensure that the status is set to MIGRATE.

13. Exit from SQL*Plus, and change directory to the new Oracle home `$ORACLE_HOME/rdbms/admin`:

    ```
    SQL> EXIT
    $ ORACLE_HOME/bin
    ```

14. Start the upgrade using the Parallel Upgrade Utility (`catctl.pl`, using the shell command `dbupgrade`), where `-d` specifies the location of the directory:

    ```
    dbupgrade -d $ORACLE_HOME/rdbms/admin
    ```

    If you do not specify any parameters, then the Parallel Upgrade Utility runs the upgrade in parallel on the number of PDBs equivalent to the number of

CPUs divided by 2. On a server with 64 CPUs, 64 divided by 2 equals 32 PDBs upgraded in parallel, carried out with two SQL processors for each PDB. CDB$ROOT remains in NORMAL mode for the duration of the upgrade.

15. Confirm that the upgrade was successful by reviewing the `upg_summary.log` If necessary, review other logs.
16. Open all PDBs, so that you can recompile the databases:

    ```
    ALTER PLUGGABLE DATABASE ALL OPEN;
    ```

17. Exit from SQL*Plus, and change directory to the new Oracle home path `$ORACLE_HOME/rdbms/admin`:

    ```
    SQL> EXIT
    cd $ORACLE_HOME/rdbms/admin
    ```

18. Run the `catcon.pl` script and the `postupgrade_fixups.sql` script that is supplied with the new release Oracle Database.

    The following example shows the command strings for running `catcon.pl`, using the `-n` parameter to specify one parallel processor for each PDB, using the `-d` parameter to specify the path where the preupgrade script that you want to run is located, using the `-l` parameter to specify the location where you want the scripts to place log files, and using the `-b` flag to specify the log file prefixes for the `postupgrade_fixups.sql` script:

    ```
    $ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -n 1
    -d \
    $ORACLE_HOME/cfgtoollogs/cdbupgr/preupgrade -l /home/oracle/
    upgrdDBA -b \
    postupgrade_fixups postupgrade_fixups.sql
    ```

19. Run `postupgrade_fixups.sql`.

    Non- CDB:

    ```
    SQL> @rdbms/admin/postupgrade_fixups.sql
    ```

    CDB:

    ```
    $ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b postupgradefixups
    -d '''.''' postupgradefixups.sql
    ```

20. Run `utlusts.sql` to verify that there are no upgrade issues.

    Non-CDB:

    ```
    SQL> @rdbms/admin/utlusts.sql
    ```

    CDB:

    ```
    $ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlu122s -d '''.'''
    utlusts.sql
    ```

When you use `catcon.pl` to run `utlusts.sql`, the log file `utlu122s0.log` is generated. The log file provides the upgrade results. You can also review the upgrade report, `upg_summary.log`.

To see information about the state of the database, run `utlusts.sql` as many times as you want, at any time after the upgrade is completed. If the `utlusts.sql` script returns errors, or shows components that do not have the status `VALID`, or if the version listed for the component is not the most recent release, then perform troubleshooting.

21. (Conditional) For Oracle RAC environments only, enter the following commands to set the initialization parameter value for `CLUSTER_DATABASE` to `TRUE`, and to start the Oracle RAC database, where *`dbname`* is the name of the Oracle RAC database:

```
ALTER SYSTEM SET CLUSTER_DATABASE=TRUE SCOPE=SPFILE;
srvctl start database -db db_unique_name
```

Your database is now upgraded.

> **Caution:**
>
> If you retain the old Oracle software, then never start the upgraded database with the old software. Only start Oracle Database using the start command in the new Oracle Database home.
>
> Before you remove the old Oracle environment, relocate any data files in that environment to the new Oracle Database environment.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for information about relocating data files

Upgrading Multitenant Architecture Sequentially Using Unplug-Plug

To upgrade pluggable databases (PDBs) that are in an earlier release multitenant container databases (CDBs), Oracle Database Release 12*c* (12.1.0.1) and later, you can unplug the PDBs from the earlier release CDB, and plug the PDBs into the later release CDB.

- About Upgrading Pluggable Databases (PDBs) Sequentially
  You can upgrade PDBs manually with the Parallel Upgrade Utility by unplugging a PDB from an earlier release CDB, plugging it into a later release CDB, and then upgrading that PDB to the later release.
- Unplugging the Earlier Release PDB from the Earlier Release CDB
  To prepare for upgrading the PDB, use this procedure to unplug the PDB from the earlier release CDB.
- Plugging in the Earlier Release PDB to the Later Release CDB
  To Plug the PDB from the earlier release CDB to the later release CDB, use the `CREATE PLUGGABLE DATABASE` command.

- Upgrading the Earlier Release PDB to the Later Release
  Open PDBs in `UPGRADE` mode use the Parallel Upgrade Utility to carry out the upgrade of the earlier-release PDB to the release level of the CDB.
- Use Inclusion or Exclusion Lists for PDB Upgrades
  If you want to upgrade a subset of earlier release PDBs, then use inclusion or exclusion lists to avoid reupgrading the CDB or PDBs that are at the new release level.

About Upgrading Pluggable Databases (PDBs) Sequentially

You can upgrade PDBs manually with the Parallel Upgrade Utility by unplugging a PDB from an earlier release CDB, plugging it into a later release CDB, and then upgrading that PDB to the later release.

You have multiple options available to you to upgrade PDBs. CDBs can contain zero, one, or more pluggable databases (PDBs). After you install a new Oracle Database release, or after you upgrade the CDB (`CDB$ROOT`), you can upgrade one or more PDBs without upgrading all of the PDBs on the CDB.

You can choose the upgrade plan that meets the needs for your service delivery. For example, you can use the AutoUpgrade utility to upgrade PDBs, or you can use the manual Parallel Upgrade Utility to upgrade PDBs individually, or with inclusion or exclusion lists. You can upgrade the CDB and all PDBs (an In Parallel manual upgrade), or you can upgrade the CDB, and then upgrade PDBs sequentially, either individually, or in sets using inclusion or exclusion lists. You can also continue to use Database Upgrade Utility (DBUA). However, the preferred option for upgrading Oracle Database is to use the AutoUpgrade utility.

If you choose to run upgrades using the Parallel Upgrade Utility to perform manual unplug-plug upgrades, then the following is a high-level list of the steps required for sequential PDB upgrades using the Parallel Upgrade Utility:

1. Unplug the earlier release PDB from the earlier release CDB.
2. Drop the PDB from the CDB.
3. Plug the earlier release PDB into the later release CDB.
4. Upgrade the earlier release PDB to a later release.

If you choose to upgrade manually using the Parallel Upgrade Utility, then you can manage PDB upgrades with lists:

- Priority lists, to set the order in which PDBs are upgraded
- Inclusion lists, which enable you to designate a set of PDBs to upgrade after the PDBs listed in the priority list are upgraded
- Exclusion lists, which enable you to designate a set of PDBs that are not upgraded

> **Note:**
>
> A PDB cannot be recovered unless it is backed up. After an upgrade using the method of creating a CDB and plugging in a PDB, be sure to back up the PDB.

Related Topics

- *Oracle Database Backup and Recovery User's Guide*
- *Oracle Database Administrator's Guide*

Unplugging the Earlier Release PDB from the Earlier Release CDB

To prepare for upgrading the PDB, use this procedure to unplug the PDB from the earlier release CDB.

1. Run the Pre-Upgrade Information Tool on the PDB.

   For example, where the PDB named `salespdb` is running in the CDB in `$ORACLE_HOME_12.2`:

   ```
   $ORACLE_HOME_12.2/jdk/bin/java -jar
   $ORACLE_HOME_19/rdbms/admin/preupgrade.jar dir /tmp -c salespdb
   ```

2. Run `preupgrade_fixups.sql` on your source database.

   For example:

   ```
   CONNECT / AS SYSDBA
   SQL> ALTER SESSION SET CONTAINER=salespdb;

   SQL> @/tmp/preupgrade_fixups_salespdb.sql
   ```

3. Follow all recommendations listed in `preupgrade.log`.
4. Close the PDB you want to unplug.

   For example, use the following command to close the PDB `salespdb`:

   ```
   SQL> ALTER PLUGGABLE DATABASE salespdb CLOSE;
   ```

5. Log back in to `CDB$ROOT`:

   ```
   CONNECT / AS SYSDBA
   SQL> ALTER SESSION SET CONTAINER=CDB$ROOT;
   ```

6. Unplug the earlier release PDB using the following SQL command syntax, where *`pdb`* is the name of the PDB, and *`path`* is the location of the PDB XML file:

   ```
   ALTER PLUGGABLE DATABASE pdb UNPLUG INTO 'path/pdb.xml';
   ```

   For example, where the pdb name is `salespdb` and path is `/home/oracle/salespdb.xml`:

   ```
   SQL> ALTER PLUGGABLE DATABASE salespdb UNPLUG INTO '/home/oracle/
   salespdb.xml';
   ```

   The following response displays when the command is completed:

   ```
   Pluggable database altered
   ```

7. Drop the pluggable database `salespdb`, but keep data files.

Oracle recommends that you drop `salespdb` after this procedure to clean up leftover information in the CDB views, and to help to avoid future issues. As a best practice guideline, back up your PDB in the destination CDB first, and then issue the `DROP` command on the source.

> **Caution:**
>
> After you drop the PDB from its original CDB, you cannot revert to it using previously taken backup, because the `DROP` command removes backup files.

To drop the pluggable database, enter the following command:

```
SQL> DROP PLUGGABLE DATABASE salespdb KEEP DATAFILES;
```

8. Exit.

Plugging in the Earlier Release PDB to the Later Release CDB

To Plug the PDB from the earlier release CDB to the later release CDB, use the `CREATE PLUGGABLE DATABASE` command.

This procedure example shows how to plug in a PDB when you are using Oracle-Managed Files. Refer to *Oracle Database Administrator's Guide* for additional information about plugging in PDBs.

1. Connect to the later release CDB.
2. Plug in the earlier release PDB using the following SQL command, where *pdb* is the name of the PDB, and *path* is the path where the PDB XML file is located:

   `CREATE PLUGGABLE DATABASE` *pdb* `USING '`*path*`/`*pdb*`.xml';`

   For example:

   ```
   SQL> CREATE PLUGGABLE DATABASE salespdb USING '/home/oracle/
   salespdb.xml';
   ```

   The following response displays when the command is completed:

   ```
   Pluggable database created.
   ```

> **Note:**
>
> When you plug in an earlier release PDB, the PDB is in restricted mode. You can only open the PDB for upgrade.

Related Topics

- *Oracle Database Administrator's Guide*

Upgrading the Earlier Release PDB to the Later Release

Open PDBs in `UPGRADE` mode use the Parallel Upgrade Utility to carry out the upgrade of the earlier-release PDB to the release level of the CDB.

1. If needed, switch to the PDB that you want to upgrade. For example, enter the following command to switch to the PDB `salespdb`:

   ```
   SQL> ALTER SESSION SET CONTAINER=salespdb;
   ```

2. Open the PDB in UPGRADE mode.

   ```
   SQL> ALTER PLUGGABLE DATABASE OPEN UPGRADE;
   ```

3. Upgrade the PDB using the Parallel Upgrade Utility command (`catctl.pl`, or the shell utility `dbupgrade`).

   When you upgrade a PDB, you use the commands you normally use with the Parallel Upgrade Utility. However, you also add the option `-c` *`PDBname`* to specify which PDB you are upgrading. Capitalize the name of your PDB as shown in the following example using the PDB named `salespdb`:

   ```
   $ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catctl.pl -d  \
   $ORACLE_HOME/rdbms/admin -c 'salespdb' -l $ORACLE_BASE catupgrd.sql
   ```

4. Review results.

   The default file path for the logs is in the path *`Oracle_base`*`/cfgtoollogs/`*`dbname/upgradedatetime`*, where *`Oracle_base`* is the Oracle base path, *`dbname`* is the database name, and *`upgradedatetime`* is the date and time for the upgrade. The date and time strings are in the character string format *YYYYMMDDHHMMSC*, in which *YYYY* designates the year, *MM* designates the month, *DD* designates the day, *HH* designates the hour, *MM* designates the minute, and *SC* designates the second.

   For example:

   ```
   $ORACLE_BASE/cfgtoollogs/salespdb/upgrade20181015120001/
   upg_summary.log
   ```

5. To execute post-upgrade fixups, and to recompile the INVALID objects in the databaseUse the utility `catcon.pl` to run the script `postupgrade_fixups.sql`:

   ```
   $ORACLE_HOME/perl/bin/perl catcon.pl –c 'salespdb' -n 1 -
   e -b postfixups -d '''.''' /tmp/cfgtoollogs/salespdb/preupgrade/
   postupgrade_fixups.sql
   ```

6. Use the utility `catcon.pl` to run `utlrp.sql` from the `$ORACLE_HOME/rdbms/admin` directory:

   ```
   $ORACLE_HOME/perl/bin/perl catcon.pl –c 'salespdb'-n 1 -e -b comp
   -d '''.''' utlrp.sql
   ```

The script recompiles `INVALID` objects in the database, and places a log file in the current directory with the name `comp0.log`.

Use Inclusion or Exclusion Lists for PDB Upgrades

If you want to upgrade a subset of earlier release PDBs, then use inclusion or exclusion lists to avoid reupgrading the CDB or PDBs that are at the new release level.

Oracle recommends that you record the containers that you upgrade, and use inclusion or exclusion lists to exclude these containers from successive bulk upgrades. Excluding upgraded containers from successive bulk upgrades ensures that the upgrade only runs on PDBs that require the upgrade. Avoiding reupgrades minimizes the overall upgrade time, and avoids unnecessary unavailability.

For example: If you have installed Oracle Database using a multitenant architecture deployment, then the containers `CDB$ROOT`, `PDB$SEED`, and any other PDBs created when the CDB was created, are part of the new release multitenant architecture. If you upgraded a CDB, and at the same time upgraded a set of PDBs to the new release, then you do not need to upgrade either the CDB containers or the upgraded PDBs again.

In either case, when you plug in earlier release PDBs and then upgrade them, upgrade the PDBs with either an exclusion list, or an inclusion list:

- Use an inclusion list to specify only the set of PDBs that you want to upgrade.
- Use an exclusion list to exclude the CDB and PDB containers that are already upgraded.

If you do not use an inclusion list or an exclusion list to limit the upgrade scope, then the Parallel Upgrade Utility (`catctl.pl`) attempts to upgrade the entire CDB, not just the PDBs that require the upgrade. During that upgrade process, your system undergoes needless downtime. The inclusion list and the exclusion list options are mutually exclusive.

Example of Manual Upgrade of Windows Non-CDB Oracle Database

These examples show the steps to complete preupgrade checks, upgrade, and postupgrade checks for an Oracle Database 11g Release 2 (11.2.0.4) upgrade to Oracle Database 19c.

The specific examples in these topics show an upgrade on the Microsoft Windows platform. They use tools specific to the Windows environment. However, these example provide an overview of a manual installation experience.

- Preparing to Upgrade Windows Non-CDB Using Command-Line Utilities
  Complete the required preupgrade steps before upgrading Non-CDB Oracle Database 11g release 2 (11.2.0.4) databases to a Non-CDB Oracle Database 12c and later releases.
- Manually Upgrading Windows Non-CDB Using Command-Line Utilities
  These examples show upgrade steps to upgrade a Non-CDB Oracle Database 11g release 2 (11.2.0.4) to a Non-CDB Oracle Database

- Running Postupgrade Fixup Scripts After Upgrading a Non-CDB Database
  These examples show the process of running the `postupgrade_fixups.sql` script for Non-CDB to Non-CDB upgrades.

Preparing to Upgrade Windows Non-CDB Using Command-Line Utilities

Complete the required preupgrade steps before upgrading Non-CDB Oracle Database 11g release 2 (11.2.0.4) databases to a Non-CDB Oracle Database 12c and later releases.

This procedure provides an example of how to use command-line utilities to prepare your system to carry out a manual upgrade of a Non-CDB Oracle Database to a current release Non-CDB Oracle Database on Windows. In this example, the Oracle Database that you are upgrading is Oracle Database 11g release 2 (11.2.0.4).

Log in as a user that is a member of the `ORA_DBA` group. Oracle recommends that you install the new release Oracle Database binaries on the server. Refer to the Oracle Database installation guide for Windows to complete that procedure.

The Preupgrade consists of the following steps:

1. Reset the user environment variables from the new release Oracle home to the 11.2.0.4 Oracle home for the Oracle Installation Owner User Account (Oracle user).
2. Test the connection to the 11.2.0.4 Oracle Database.
3. Run the Pre-Upgrade Information Tool on the 11.2.0.4 Oracle Database home.
4. Review the Pre-Upgrade Information Tool log (`preupgrade.log`). If there are `preupgrade_fixups.sql` scripts that you must run, then run them. In this example, we show manual fixes for the following errors:
   - Oracle Enterprise Manager is present (`em_present`).
   - The OLAP catalog is present (`amd_exists`).
   - The ADMINISTER DATABASE TRIGGER privilege must be granted (`trgowner_no_admndbtrg`).
   - Materialized views are not refreshed (`mv_refresh`).
   - Oracle Application Express is not upgraded (`apex_upgrade_msg`)
5. Run `preupgrade_fixups.sql` scripts.
6. Back up your database.

These examples use the following systems, paths, and users:

Oracle Database 11g release 2 (11.2.0.4)

- Oracle Installation Owner user account: `oracle1`, which is a member of the ORA_DBA group.
- Oracle home: `C:\app\oracle\product\11.2.0\dbhome_1`

Oracle Database 19c

- Oracle Installation Owner user account: `oracle2`, which is a member of the ORA_DBA group.

- Oracle home: `C:\app\oracle1\product\19\dbhome_1`

Example 4-18 Resetting the User Environment Variables

This example shows how to change the environment variables from values set for the Oracle Database 19c installation:

```
C:\app\oracle1\product\19\dbhome_1\rdbms\admin>set
ORACLE_HOME=C:\app\oracle2\product\11.2.0\dbhome_1
C:\app\oracle1\product\19\dbhome_1\rdbms\admin>set ORACLE_SID=orcl11
```

Example 4-19 Testing the Connection to the Database

This example shows how to test the connection to the database, and ensure that you are connecting to the release 11.2.0.4 database that you want to upgrade.

```
C:\app\oracle1\product\19\dbhome_1\rdbms\admin>sqlplus / as sysdba

    SQL*Plus: Release 11.2.0.4.0 Production on Tue Dec 11 12:14:49 2018

    Copyright (c) 1982, 2011, Oracle.  All rights reserved.


    Connected to:
    Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit
Production
    With the Partitioning, OLAP, Data Mining and Real Application
Testing options

    SQL> select version from v$instance;

    VERSION
    -----------------
    11.2.0.4.0

    SQL>
```

Example 4-20 Running the Pre-Upgrade Information Tool (preupgrade.jar)

This example shows how to run the tool using the command `java -jar %ORACLE_HOME19%\rdbms\admin\preupgrade.jar`.

The Preupgrade Information Tool prints the path to the log file, and prints the path to the fixup scripts. By default, these fixup scripts are placed in `%ORACLE_BASE%`.

```
C:\app\oracle1\product\19\dbhome_1\rdbms\admin>java -jar preupgrade.jar

Preupgrade generated files:
        C:\app\oracle2\cfgtoollogs\orcl11\preupgrade\preupgrade.log

C:\app\oracle2\cfgtoollogs\orcl11\preupgrade\preupgrade_fixups.sql

C:\app\oracle2\cfgtoollogs\orcl11\preupgrade\postupgrade_fixups.sql
C:\app\oracle1\product\12.2.0\dbhome_1\rdbms\admin>
```

Example 4-21 Reviewing the Pre-Upgrade Information Tool Log (preupgrade.log)

In this example, the log file `preupgrade.log` is generated in the following path: `C:\app\oracle2\cfgtoollogs\orcl11\preupgrade\preupgrade.log`.

The log file can direct you to run some SQL fixup files manually. If it directs you to run SQL files, then copy these files to the earlier release 11.2.0.4 Oracle home, and then run the scripts.

Example 4-22 Running the Preupgrade Fixup SQL script (preupgrade_fixups.sql)

```
C:\app\oracle1\product\19\dbhome_1\rdbms\admin>sqlplus / as sysdba
@C:\app\oracle2\cfgtoollogs\orcl11\preupgrade\preupgrade_fixups.sql

    SQL*Plus: Release 11.2.0.4.0 Production on Tue Dec 11 12:34:08 2018

    Copyright (c) 1982, 2011, Oracle.  All rights reserved.


    Connected to:
    Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit
Production
    With the Partitioning, OLAP, Data Mining and Real Application
Testing options

    Executing Oracle PRE-Upgrade Fixup Script

    Auto-Generated by:       Oracle Preupgrade Script
                             Version: 19.0.0.0 Build: 1
    Generated on:            2018-12-11 12:18:06

    Source Database:         ORCL11
    Source Database Version: 11.2.0.4.0
    For Upgrade to Version:  12.2.0.1.0
                              Fixup
    Check Name                Status  Further DBA Action
    ----------                ------  ------------------
    em_present                Failed  Manual fixup recommended.
    amd_exists                Failed  Manual fixup recommended.
    dictionary_stats          Failed  Manual fixup recommended.
    trgowner_no_admndbtrg     Failed  Manual fixup recommended.
    mv_refresh                Failed  Manual fixup recommended.
    apex_upgrade_msg          Failed  Manual fixup recommended.

    PL/SQL procedure successfully completed.
```

In this example, some automatic checks failed, and required manual fixups. Refer to the examples that follow to see how to perform manual fixups.

Example 4-23 Manually Removing Oracle Enterprise Manager (em_present)

The `preupgrade_fixups.sql` log file provides you with instructions to remove Oracle Enterprise Manager manually (`em_present`):

```
+ Remove the EM repository.

         - Copy the rdbms/admin/emremove.sql script from the target
19.0.0.0.0
         ORACLE_HOME into the source 11.2.0.4.0 ORACLE_HOME.

         Step 1: If database control is configured, stop EM Database
Control,
         using the following command

           $> emctl stop dbconsole

         Step 2: Connect to the database using the SYS account AS SYSDBA

           SET ECHO ON;
           SET SERVEROUTPUT ON;

          @emremove.sql
        ...
        ...
        ...

        Execution:

        Step 1 - Stop dbconsole:

            C:\app\oracle1\product\19\dbhome_1\rdbms\admin> emctl stop
dbconsole

            Oracle Enterprise Manager 11g Database Control Release
11.2.0.4.0
            Copyright (c) 1996, 2011 Oracle Corporation.  All rights
reserved.
            https://db01.example.com:1158/em/console/aboutApplication
            The OracleDBConsoleorcl11 service is stopping.....
            The OracleDBConsoleorcl11 service was stopped successfully.

        Step 2 - Remove em (provide credentials if needed)

            ...
            ...
            ...

            Dropping synonym : SETEMVIEWUSERCONTEXT ...
            Dropping synonym : SMP_EMD_AVAIL_OBJ ...
            Dropping synonym : SMP_EMD_DELETE_REC_ARRAY ...
            Dropping synonym : SMP_EMD_INTEGER_ARRAY ...
            Dropping synonym : SMP_EMD_INTEGER_ARRAY_ARRAY ...
            Dropping synonym : SMP_EMD_NVPAIR ...
            Dropping synonym : SMP_EMD_NVPAIR_ARRAY ...
```

```
            Dropping synonym : SMP_EMD_STRING_ARRAY ...
            Dropping synonym : SMP_EMD_STRING_ARRAY_ARRAY ...
            Dropping synonym : SMP_EMD_TARGET_OBJ ...
            Dropping synonym : SMP_EMD_TARGET_OBJ_ARRAY ...
            Finished phase 5
            Starting phase 6 : Dropping Oracle Enterprise Manager
related other roles ...
            Finished phase 6
            The Oracle Enterprise Manager related schemas and objects
are dropped.
            Do the manual steps to studown the DB Control if not done
before running this
            script and then delete the DB Control configuration files

            PL/SQL procedure successfully completed.

            SQL>
```

Example 4-24 Manually Removing the OLAP Catalog (amd_exists)

The `preupgrade_fixups.sql` log file provides you with instructions to remove the OLAP catalog manually (`amd_exists`):

```
 + Remove OLAP Catalog by running the 11.2.0.4.0 SQL script
     $ORACLE_HOME/olap/admin/catnoamd.sql script.

     The OLAP Catalog component, AMD, exists in the database.

     sqlplus / as sysdba @%ORACLE_HOME%\olap\admin\catnoamd.sql
        ...
        ...
        ...

        Synonym dropped.

        User dropped.

        Role dropped.

        SQL>
```

Example 4-25 Gathering Current Dictionary Statistics (dictionary_stats)

The `preupgrade_fixups.sql` log file provides you with instructions to gather directory statistics (`dictionary_stats`).

```
       + Gather dictionary statistics prior to database upgrade in off-
peak time
     using:

       EXECUTE dbms_stats.gather_dictionary_stats;

        SQL> EXECUTE dbms_stats.gather_dictionary_stats;
```

```
PL/SQL procedure successfully completed.
```

Example 4-26 Granting the ADMINISTER DATABASE TRIGGER privilege (trgowner_no_admndbtrg)

The `preupgrade_fixups.sql` log file provides you with instructions to grant the ADMINISTER DATABASE TRIGGER privilege (`trgowner_no_admndbtrg`):

```
+ Directly grant ADMINISTER DATABASE TRIGGER privilege to the owner of
the
         trigger or drop and re-create the trigger with a user that was
granted
         directly with such. You can list those triggers using "SELECT
OWNER,
         TRIGGER_NAME FROM DBA_TRIGGERS WHERE
BASE_OBJECT_TYPE='DATABASE' AND
         OWNER NOT IN (SELECT GRANTEE FROM DBA_SYS_PRIVS WHERE
         PRIVILEGE='ADMINISTER DATABASE TRIGGER')"

            SQL> SELECT OWNER,
              2       TRIGGER_NAME FROM DBA_TRIGGERS WHERE
BASE_OBJECT_TYPE='DATABASE' AND
              3       OWNER NOT IN (SELECT GRANTEE FROM DBA_SYS_PRIVS
WHERE
              4       PRIVILEGE='ADMINISTER DATABASE TRIGGER');

            OWNER                          TRIGGER_NAME
            ------------------------------
------------------------------
            MDSYS                          SDO_DROP_USER
            MDSYS                          SDO_ST_SYN_CREATE
            MDSYS                          SDO_TOPO_DROP_FTBL
            MDSYS                          SDO_GEOR_BDDL_TRIGGER
            MDSYS                          SDO_GEOR_ADDL_TRIGGER
            MDSYS                          SDO_NETWORK_DROP_USER

            6 rows selected.

            SQL> grant ADMINISTER DATABASE TRIGGER to MDSYS;

            Grant succeeded.
```

Example 4-27 Refreshing Materialized Views (mv_refresh)

The `preupgrade_fixups.sql` log file provides you with instructions to refresh materialized views (`mv_refresh`):

```
+ Please make sure that all the MVs are refreshed and sys.sumdelta$
         becomes empty before doing upgrade, unless you have strong
business
         reasons not to do so. You can use dbms_mview.refresh() to
refresh the
         MVs except those stale ones  to be kept due to business need.
```

```
If there
         are any stale MVs depending on changes in sys.sumdelta$, do
not truncate
         it, because doing so will cause wrong results after
refresh.

            SQL> declare
              2  num_failures integer(3) :=0;
              3  begin
              4  DBMS_MVIEW.REFRESH_ALL_MVIEWS(num_failures,'C','',
TRUE, FALSE);
              5  end;
              6  /

            PL/SQL procedure successfully completed.

            SQL> select count(1) from sumdelta$;

              COUNT(1)
            ----------
                     0

            SQL>
```

Example 4-28 Upgrading Oracle Application Express (apex_upgrade_msg)

```
preupgrade_fixups.sqlapex_upgrade_msg

+ Consider upgrading APEX manually, before the database upgrade.

         The database contains APEX version 3.2.1.00.12 and will need
to be
         upgraded to at least version 5.0.4.00.11.

         To reduce database upgrade time, you can upgrade APEX manually
before
         the database upgrade.  Refer to My Oracle Support Note
1088970.1 for
         information on APEX installation upgrades.

        Download: http://www.oracle.com/technetwork/developer-tools/
apex/downloads/index.html provide credentials.

        Then go to: http://www.oracle.com/technetwork/developer-tools/
apex/application-express/upgrade-apex-for-xe-154969.html and
                    https://docs.oracle.com/cd/E59726_01/install.50/
e39144/toc.htm to guide you to do the upgrade.

        Once downloaded, please go to the directory and unzip the
files, run the apex upgrade from there.

        ...
        ...
            Completing registration process. 12:07:41
            Validating installation.  12:07:41
```

```
            ...Starting validation 12:07:41
            ...Database user "SYS", database schema "APEX_050000",
user# "90" 12:07:41
            ...272 packages
            ...265 package bodies
            ...465 tables
            ...8 functions
            ...16 procedures
            ...4 sequences
            ...497 triggers
            ...1582 indexes
            ...255 views
            ...0 libraries
            ...14 types
            ...5 type bodies
            ...0 operators
            ...0 index types
            ...Begin key object existence check 12:07:53
            ...Completed key object existence check 12:07:54
            ...Setting DBMS Registry 12:07:54
            ...Setting DBMS Registry Complete 12:07:54
            ...Exiting validate 12:07:54

            PL/SQL procedure successfully completed.

            timing for: Validate Installation
            Elapsed: 00:00:13.00

            Session altered.

            timing for: Complete Installation
            Elapsed: 00:18:40.49

        ...
        ...
```

> **Caution:**
>
> After you complete preupgrade steps, Oracle recommends that you back up your database before proceeding with the upgrade.

Related Topics

- Backing Up Oracle Database for Upgrading

Manually Upgrading Windows Non-CDB Using Command-Line Utilities

These examples show upgrade steps to upgrade a Non-CDB Oracle Database 11g release 2 (11.2.0.4) to a Non-CDB Oracle Database

After you complete running preupgrade steps, you can upgrade the Non-CDB Oracle Database to the new release Non-CDB Oracle Database. Before starting the upgrade,

you must stop the database services. You can stop database services either by using command-line commands, or by using Microsoft Windows PowerShell scripting.

The sequence of steps to complete the upgrade is as follows:

1. Stop the database service, using either command-line commands, or PowerShell.
2. Delete the database service from the earlier release Oracle home.
3. Stop the listener from the earlier release Oracle home
4. Set the environment variables to the new Oracle home.
5. Copy database files, such as `tnsnames.ora`, `listener.ora`, password files, wallets, and other similar files to the new Oracle home.
6. Copy the PFILE to the new Oracle Database Oracle home, and create a new service using the Oracle Database binary in the new Oracle home. (In this example, we assume that the PFILE is compatible. It is possible that your PFILE is not compatible with the new release.)
7. Start the database upgrade.
8. Complete the post-upgrade steps.

Example 4-29 Stopping the Database Service Using Command-Line Commands

1. If you do not know the service name, then identify the service name.

```
c:\apex\apex>sc query type= service | find /i "orcl11"
                SERVICE_NAME: OracleServiceORCL11
                DISPLAY_NAME: OracleServiceORCL11
```

2. Using the service name, find out what the status is of the service.

```
c:\apex\apex>sc query  OracleServiceORCL11

                SERVICE_NAME: OracleServiceORCL11
                    TYPE               : 10  WIN32_OWN_PROCESS
                    STATE              : 4  RUNNING
                                            (STOPPABLE, PAUSABLE,
ACCEPTS_SHUTDOWN)
                    WIN32_EXIT_CODE    : 0  (0x0)
                    SERVICE_EXIT_CODE  : 0  (0x0)
                    CHECKPOINT         : 0x0
                    WAIT_HINT          : 0x0
```

3. Stop the service

```
c:\apex\apex>sc stop  OracleServiceORCL11

                SERVICE_NAME: OracleServiceORCL11
                    TYPE               : 10  WIN32_OWN_PROCESS
                    STATE              : 3  STOP_PENDING
                                            (STOPPABLE, PAUSABLE,
ACCEPTS_SHUTDOWN)
                    WIN32_EXIT_CODE    : 0  (0x0)
                    SERVICE_EXIT_CODE  : 0  (0x0)
```

```
                    CHECKPOINT         : 0x5
                    WAIT_HINT          : 0x15f90
```

4. Wait for a few minutes, and then check the status.

```
c:\apex\apex>sc query  OracleServiceORCL11

                SERVICE_NAME: OracleServiceORCL11
                    TYPE               : 10  WIN32_OWN_PROCESS
                    STATE              : 1  STOPPED
                    WIN32_EXIT_CODE    : 0  (0x0)
                    SERVICE_EXIT_CODE  : 0  (0x0)
                    CHECKPOINT         : 0x0
                    WAIT_HINT          : 0x0
```

After the services are stopped, you can proceed to delete the services.

Example 4-30 Stopping the Database Service Using Microsoft Windows PowerShell Scripting

1. Check the status of the service.

```
PS C:\app\oraclbm\cfgtoollogs\orcl\preupgrade> Get-Service | Where-
Object {$_.displayName.Contains("ORCL11")}

                Status   Name               DisplayName
                ------   ----               -----------
                Stopped  OracleJobSchedu... OracleJobSchedulerORCL11
                Running  OracleServiceOR... OracleServiceORCL11
                Running  OracleVssWriter... Oracle ORCL11 VSS
Writer Service
```

2. Stop the service.

```
PS C:\app\oraclbm\cfgtoollogs\orcl\preupgrade> Get-Service | Where-
Object {$_.displayName.Contains("ORCL11")} | Stop-Ser
                vice
                WARNING: Waiting for service 'OracleServiceORCL11
(OracleServiceORCL11)' to stop...
                WARNING: Waiting for service 'OracleServiceORCL11
(OracleServiceORCL11)' to stop...
                WARNING: Waiting for service 'OracleServiceORCL11
(OracleServiceORCL11)' to stop...
                WARNING: Waiting for service 'OracleServiceORCL11
(OracleServiceORCL11)' to stop...
                WARNING: Waiting for service 'OracleServiceORCL11
(OracleServiceORCL11)' to stop...
                WARNING: Waiting for service 'OracleServiceORCL11
(OracleServiceORCL11)' to stop...
                WARNING: Waiting for service 'OracleServiceORCL11
(OracleServiceORCL11)' to stop...
                WARNING: Waiting for service 'OracleServiceORCL11
(OracleServiceORCL11)' to stop...
                WARNING: Waiting for service 'OracleServiceORCL11
```

```
(OracleServiceORCL11)' to stop...
                WARNING: Waiting for service 'OracleServiceORCL11
(OracleServiceORCL11)' to stop...
                WARNING: Waiting for service 'OracleServiceORCL11
(OracleServiceORCL11)' to stop...
```

After the services are stopped, you can proceed to delete the database service.

Example 4-31 Deleting the Database Service from the Earlier Release Oracle Home

For Oracle Database 11g release 2 (11.2.0.4) on Windows, use ORADIM from the earlier release Oracle home. ORADIM is a Windows-specific utility that you can use to administer the Windows service.

```
c:\apex\apex>oradim -delete -sid orcl11
            Unable to stop service, OS Error = 1062
            Instance deleted.
```

Example 4-32 Stopping the Listener for the Earlier Release Oracle Home

```
c:\apex\apex>lsnrctl status

            LSNRCTL for 64-bit Windows: Version 11.2.0.4.0 - Production
on 13-JUL-2016 13:58:52

            Copyright (c) 1991, 2012, Oracle.  All rights reserved.

            Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)
(KEY=EXTPROC1522)))
            STATUS of the LISTENER
            ------------------------
            Alias                     LISTENER
            Version                   TNSLSNR for 64-bit Windows:
Version 11.2.0.4.0 - Production
            Start Date                05-JUL-2018 10:01:30
            Uptime                    8 days 3 hr. 57 min. 23 sec
            Trace Level               off
            Security                  ON: Local OS Authentication
            SNMP                      OFF
            Listener Parameter File
C:\app\oracle2\product\11.2.0\dbhome_1\network\admin\listener.ora
            Listener Log File
C:\app\oracle2\diag\tnslsnr\slc01auu\listener\alert\log.xml
            Listening Endpoints Summary...
              (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(PIPENAME=\
\.\pipe\EXTPROC1522ipc)))
              (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=db01.example.com)(PORT=1522)))
            Services Summary...
            Service "CLRExtProc" has 1 instance(s).
              Instance "CLRExtProc", status UNKNOWN, has 1 handler(s)
for this service...
            The command completed successfully
```

```
c:\apex\apex>lsnrctl stop

            LSNRCTL for 64-bit Windows: Version 11.2.0.4.0 - Production
on 13-JUL-2018 13:59:00

            Copyright (c) 1991, 2012, Oracle.  All rights reserved.

            Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)
(KEY=EXTPROC1522)))
            The command completed successfully
```

Example 4-33 Setting the environment variables to the new Oracle home

```
c:\apex\apex>set ORACLE_HOME=C:\app\oracle1\product\18.1.0\dbhome_1

c:\apex\apex>SET PATH=%ORACLE_HOME%\BIN;%PATH%

c:\apex\apex>SET ORACLE_SID=ORCL11
```

Example 4-34 Copying the PFILE to the New Oracle home, and Creating a New Service Using the New Oracle Database binary

This example shows how to use the Windows-specific ORADIM utility to create a new database service. In this example, the PFILE from the earlier release is compatible with the new release.

```
c:\apex\apex>C:\app\oracle1\product\18.1.0\dbhome_1\bin\oradim -new -
sid orcl11 -syspwd
My-sys-password -maxusers 1000 -startmode auto -pfile
C:\app\oracle1\product\18.1.0\dbhome_1\database\initorcl11.ora

            Enter password for Oracle service user:
            Instance created.
```

Example 4-35 Starting the Database Upgrade

Start the upgrade using the Parallel Upgrade Utility.

Note:

You can find that you have to change the folder symbolic links on the PFILE for paths such as DIAGNOSTIC_DEST. If this is the case, then you see errors such as the following examples:

```
ORA-48173: error checking directory existence during ADR
initialization [C:\app\oracle2\diag\rdbms\orcl11]
ORA-00205: error in identifying control file, check alert log
for more info
```

If you see these errors, then grant permissions to the user on the earlier release Oracle home, and on all the related folders with inheritance. Make this change also on any access control list (ACL) that the database accesses, such as wallet files. This step is particularly necessary if you are using the earlier release Oracle home as an ORADATA location.

If you are using the built-in account, then the service runs as `LocalSystem`. If you are using a virtual account, then services run using the Windows virtual accounts, with names derived from the service name. If the Oracle Database 19c binaries owner is a virtual account, then you must add that virtual account to the group `ORA_`*`Homename`*`_SVCAACCTS`, instead of adding the Oracle Home User.

Oracle Database 19c includes a shell script, `dbupgrade`. The shell script calls the Parallel Upgrade Utility (`catctl.pl`), so that you can run the upgrade as a command. This example shows the upgrade of Oracle Database from release 11.2.0.4 to release 19c. The upgrade time for your system can vary from the results in this example.

```
C:\app\oracle1\product\19.1.0\dbhome_1\bin>dbupgrade
            ...
            ...
            ...
                C:\app\oracle1\product\19.1.0\dbhome_1\bin>REM Batch
file to execute catctl.pl

                Argument list for
[C:\app\oracle1\product\19\dbhome_1\rdbms\admin\catctl.pl]
                Run in                c = 0
                Do not run in         C = 0
                Input Directory       d = 0
                Echo OFF              e = 1
                Simulate              E = 0
                Forced cleanup        F = 0
                Log Id                i = 0
                Child Process         I = 0
                Log Dir               l = c:\temp2
                Priority List Name    L = 0
                Upgrade Mode active   M = 0
                SQL Process Count     n = 4
                SQL PDB Process Count N = 0
                Open Mode Normal      o = 0
                Start Phase           p = 0
```

```
                End Phase             P = 0
                Reverse Order         r = 0
                AutoUpgrade Resume    R = 0
                Script                s = 0
                Serial Run            S = 0
                RO User Tablespaces   T = 0
                Display Phases        y = 0
                Debug catcon.pm       z = 0
                Debug catctl.pl       Z = 0

                catctl.pl VERSION: [19.0.0.0.0]
                    STATUS: [production]
                    BUILD: [RDBMS_MAIN_WINDOWS.X64_160624]


C:\app\oracle1\product\19.1.0\dbhome_1\rdbms\admin\orahome.exe =
[C:\app\oracle1\product\19.1.0\dbhome_1]

C:\app\oracle1\product\19.1.0\dbhome_1\bin\orabasehome.exe =
[C:\app\oracle1\product\189.1.0\dbhome_1]
                catctlGetOrabase =
[C:\app\oracle1\product\19.1.0\dbhome_1]

                Analyzing file
C:\app\oracle1\product\19.1.0\dbhome_1\rdbms\admin\catupgrd.sql

                Log file directory = [c:\temp2]

                catcon: ALL catcon-related output will be written to
[c:\temp2/catupgrd_catcon_3252.lst]
                catcon: See [c:\temp2/catupgrd*.log] files for output
generated by scripts
                catcon: See [c:\temp2/catupgrd_*.lst] files for spool
files, if any

                Number of Cpus        = 2
                Database Name         = orcl11
                DataBase Version      = 11.2.0.4.0
                Parallel SQL Process Count            = 4
                Components in [orcl11]
                    Installed [APEX APS CATALOG CATJAVA CATPROC CONTEXT
JAVAVM ORDIM OWM SDO XDB XML XOQ]
                Not Installed [DV EM MGW ODM OLS RAC WK]

                ------------------------------------------------------
                Phases [0-109]         Start Time:[2018_12_13 15:49:40]
                ------------------------------------------------------
                ***********   Executing Change Scripts   ***********
                Serial   Phase #:0    [orcl11] Files:1

            ...
            ...
            ...
                ------------------------------------------------------
```

```
Phases [0-109]         End Time:[2018_12_13 16:53:27]
--------------------------------------------------

Grand Total Time: 3830s

 LOG FILES: (c:\temp2\catupgrd*.log)

Upgrade Summary Report Located in:
c:\temp2\upg_summary.log

Grand Total Upgrade Time:    [0d:1h:3m:50s]
```

Example 4-36 Completing the Post-Upgrade Checks

Start the upgraded Oracle Database, and check the version and status.

```
C:\app\oracle1\product\19.1.0\dbhome_1\bin>sqlplus / as sysdba

SQL*Plus: Release 19.1.0 Production on Thu Dec 13
09:38:50 201

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> startup
ORACLE instance started.

Total System Global Area 1828716544 bytes
Fixed Size                  8740096 bytes
Variable Size             503317248 bytes
Database Buffers         1308622848 bytes
Redo Buffers                8036352 bytes
Database mounted.
Database opened.
SQL> select open_mode from v$database;

OPEN_MODE
--------------------
READ WRITE

SQL> select version from v$instance;

VERSION
-----------------
19.0.0.0.0

SQL>
```

After the upgrade is complete, carry out post-upgrade checks to ensure that objects are valid, and that there are no remaining issues.

Running Postupgrade Fixup Scripts After Upgrading a Non-CDB Database

These examples show the process of running the `postupgrade_fixups.sql` script for Non-CDB to Non-CDB upgrades.

The examples show an upgrade from a Non-CDB Oracle Database 11g release 2 (11.2.0.4) to a Non-CDB Oracle Database 19c.

Example 4-37 Running the Postupgrade Fixups Script

Run the `postupgrade_fixups.sql` script that you generated with the Pre-Upgrade Information Tool in the earlier release Oracle home before starting the upgrade:

```
SQL> @C:\app\oracle\cfgtoollogs\orcl11\preupgrade\postupgrade_fixups.sql

                  SQL> REM

                  SQL> REM    Oracle POST-Upgrade Fixup Script

                  SQL> REM

                  SQL> REM    Auto-Generated by: Oracle Preupgrade
Script

                  SQL> REM    Version: 19.0.0.0.0

                  SQL> REM    Generated on:            2018-12-05
12:18:07

                  SQL> REM

                  SQL> REM    Source Database:         ORCL11

                  SQL> REM    Source Database Version: 11.2.0.4.0

                  SQL> REM    For Upgrade to Version:    19.0.0.0.0

                  SQL> REM

                  SQL>

                  SQL> REM

                  SQL> REM    Setup Environment

                  SQL> REM

                  SQL> SET ECHO OFF SERVEROUTPUT ON FORMAT WRAPPED TAB
OFF LINESIZE 200;

                  Executing Oracle POST-Upgrade Fixup Script

                  Auto-Generated by:       Oracle Preupgrade Script
```

```
          Version: 19.0.0.0.0

                   Generated on:            2018-12-05 12:18:07


                   Source Database:         ORCL11

                   Source Database Version: 11.2.0.4.0

                   For Upgrade to Version:  19.0.0.0.0

          Fixup

                      Check Name                Status  Further DBA
Action

                      ----------                ------
------------------

                      depend_usr_tables         Failed  Manual fixup
recommended.

                      old_time_zones_exist      Failed  Manual fixup
recommended.

                      post_dictionary           Failed  Manual fixup
recommended.

                      fixed_objects             Passed  None

                      upg_by_std_upgrd          Passed  None

                      PL/SQL procedure successfully completed.

                      Elapsed: 00:00:11.26

 SQL>
```

In this example, the `postupgrade_fixup.sql` script could not complete three fixes. so you are instructed to carry out manual fixups. This outcome is normal. To carry out the manual fixups, look at the log file, `preupgrade.log`. It contains instructions and script information for how to run manual fixup scripts. These fixup scripts are generated for you by the `postupgrade_fixups.sql` script. Follow the instructions provided in the log file to fix the issues.

Example 4-38 Manual Fixup Instructions for Oracle-Maintained Types In User Tables (depend_usr_tables)

```
    + If you use the -T option for the database upgrade, then run

                    @?/rdbms/admin/utluptabdata.sql after the upgrade
is complete, to
```

```
                     VALIDATE and UPGRADE any user tables affected by
changes to

                     Oracle-Maintained
types.


  SQL>
@C:\app\oracle\product\19.1.0\dbhome_1\rdbms\admin\utluptabdata.sql

  PL/SQL procedure successfully completed.

                     SQL>
```

Example 4-39 Manual Fixup Instructions for Time Zone Version (old_time_zones_exist)

```
+ Upgrade the database time zone version using the DBMS_DST package.


                     The database is using timezone datafile version 14
and the target

                     19.0.0.0.0 database ships with timezone datafile
version 26.

                     Oracle recommends using the most recent timezone
data.  For further

                     information, refer to My Oracle Support Note
1585343.1


                     SQL>
@C:\Users\oracle\Downloads\DBMS_DST_scriptsV1.9\DBMS_DST_scriptsV1.9\cou
ntstatsTSTZ.sql

                     .

                     Amount of TSTZ data using num_rows stats info in
DBA_TABLES.

                     .

                     For SYS tables first...

                     Note: empty tables are not listed.

                     Stat date  - Owner.Tablename.Columnname - num_rows

                     05/07/2019 - SYS.AQ$_ALERT_QT_S.CREATION_TIME - 5

                     ...
```

```
                    Total numrow of SYS TSTZ columns is : 14652

                    There are in total 154 non-SYS TSTZ columns.

                    .

                    ...

                    For non-SYS tables ...

                    Note: empty tables are not listed.

                    ...

                    ...

                    Stat date  - Owner.Tablename.Columnname - num_rows

                    Total numrow of non-SYS TSTZ columns is : 17

                    There are in total 32 non-SYS TSTZ columns.

                    Total Minutes elapsed : 0


SQL>

                    SQL> spool upg_tzv_check.log

                    SQL>
@C:\Users\oracle\Downloads\DBMS_DST_scriptsV1.9\DBMS_DST_scriptsV1.9\upg
_tzv_check.sql

                    INFO: Starting with RDBMS DST update preparation.

                    INFO: NO actual RDBMS DST update will be done by
this script.

                    INFO: If an ERROR occurs the script will EXIT
sqlplus.

                    INFO: Doing checks for known issues ...

                    INFO: Database version is 19.0.0.0.0 .

                    INFO: Database RDBMS DST version is DSTv14 .

                    INFO: No known issues detected.

                    INFO: Now detecting new RDBMS DST version.

                    A prepare window has been successfully started.

                    INFO: Newest RDBMS DST version detected is DSTv26 .
```

```
                    INFO: Next step is checking all TSTZ data.

                    INFO: It might take a while before any further
output is seen ...

                    A prepare window has been successfully ended.

                    INFO: A newer RDBMS DST version than the one
currently used is found.

                    INFO: Note that NO DST update was yet done.

                    INFO: Now run upg_tzv_apply.sql to do the actual
RDBMS DST update.

                    INFO: Note that the upg_tzv_apply.sql script will

                    INFO: restart the database 2 times WITHOUT any
confirmation or prompt.

                    SQL> spool
off


                    SQL>
@C:\Users\oracle\Downloads\DBMS_DST_scriptsV1.9\DBMS_DST_scriptsV1.9\upg
_tzv_apply.sql

                    INFO: If an ERROR occurs the script will EXIT
sqlplus.

                    INFO: The database RDBMS DST version will be updated
to DSTv26 .

                    WARNING: This script will restart the database 2
times

                    WARNING: WITHOUT asking ANY confirmation.

                    WARNING: Hit control-c NOW if this is not intended.

                    INFO: Restarting the database in UPGRADE mode to
start the DST upgrade.

                    Database closed.

                    Database dismounted.  <--------- you might need to
wait a couple of minutes and hit enter a couple of times.

                    ORACLE instance shut down.

                    ORACLE instance started.
```

```
                    Total System Global Area 1828716544 bytes

                    Fixed Size                  8740096 bytes

                    Variable Size             503317248 bytes

                    Database Buffers         1308622848 bytes

                    Redo Buffers                8036352 bytes

                    Database mounted.

                    Database opened.

                    INFO: Starting the RDBMS DST upgrade.

                    INFO: Upgrading all SYS owned TSTZ data.

                    INFO: It might take time before any further output is
seen ... <--------- you might need to wait a couple of minutes
                    and hit enter a couple of times.

                    An upgrade window has been successfully started.

                    INFO: Restarting the database in NORMAL mode to
upgrade non-SYS TSTZ data.

                    Database closed.

                    Database dismounted.

                    ORACLE instance shut down.

                    ORACLE instance started.


                    Total System Global Area 1828716544 bytes

                    Fixed Size                  8740096 bytes

                    Variable Size             503317248 bytes

                    Database Buffers         1308622848 bytes

                    Redo Buffers                8036352 bytes

                    Database mounted.

                    Database opened.   ... <--------- you might need to
wait a couple of minutes and hit enter a couple of times.

                    INFO: Upgrading all non-SYS TSTZ data.

                    INFO: It might take time before any further output is
seen ...
```

```
                    INFO: Do NOT start any application yet that uses TSTZ
data! ... <--------- you might need to wait a couple of minutes
                    and hit enter a couple of times.

                    INFO: Next is a list of all upgraded tables:

                    Table list: "IX"."AQ$_ORDERS_QUEUETABLE_L"

                    Number of failures: 0

                    Table list: "IX"."AQ$_ORDERS_QUEUETABLE_S"

                    Number of failures: 0

                    Table list: "IX"."AQ$_STREAMS_QUEUE_TABLE_L"

                    Number of failures: 0

                    Table list: "IX"."AQ$_STREAMS_QUEUE_TABLE_S"

                    Number of failures: 0

                    Table list: "APEX_050000"."WWV_FLOW_DEBUG_MESSAGES"

                    Number of failures: 0

                    Table list: "APEX_050000"."WWV_FLOW_DEBUG_MESSAGES2"

                    Number of failures: 0

                    Table list: "APEX_050000"."WWV_FLOW_FEEDBACK"

                    Number of failures: 0

                    Table list: "APEX_050000"."WWV_FLOW_FEEDBACK_FOLLOWUP"

                    Number of failures: 0

                    Table list: "APEX_050000"."WWV_FLOW_WORKSHEET_NOTIFY"

                    Number of failures: 0

                    Table list:
"GSMADMIN_INTERNAL"."AQ$_CHANGE_LOG_QUEUE_TABLE_L"

                    Number of failures: 0

                    Table list:
"GSMADMIN_INTERNAL"."AQ$_CHANGE_LOG_QUEUE_TABLE_S"

                    Number of failures: 0

                    INFO: Total failures during update of TSTZ data: 0 .
```

```
                    An upgrade window has been successfully ended.

                    INFO: Your new Server RDBMS DST version is DSTv26 .

                    INFO: The RDBMS DST update is successfully finished.

                    INFO: Make sure to exit this sqlplus session.

                    INFO: Do not use it for timezone related selects.

               SQL>
```

Example 4-40 Manual Fixup for Refreshing Dictionary Statistics (post_dictionary)

```
+ Gather dictionary statistics after the upgrade using the command:

                EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;

                This recommendation is given for all preupgrade
runs.

         SQL> EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;

         PL/SQL procedure successfully completed.


SQL>
```

About Dbupgrade Scripts and catupgrd.sql in Earlier Releases of Oracle Database

The function of the `catupgrd.sql` script is replaced by the Parallel Upgrade Utility, `catctl.pl`, and the `dbupgrade` and `dbupgrade.cmd` scripts.

In earlier releases of Oracle Database, the `catupgrd.sql` Upgrade Utility processed the upgrade. Starting with Oracle Database 12c release 1 (12.1), this script is replaced by the `catctl.pl` Parallel Upgrade Utility, and its command-line script, `dbupgrade`. The Parallel Upgrade Utility provides both parallel processing mode and serial modes.

The `dbupgrade` script calls `catctl.pl` to create and alter a set of data dictionary tables. The upgrade scripts also call `catctl.pl` to upgrade or install the following database components in the new Oracle Database 12c database:

- Oracle Database Catalog Views
- Oracle Database Packages and Types
- JServer JAVA Virtual Machine

- Oracle Database Java Packages
- Oracle XDK
- Oracle Real Application Clusters
- Oracle Workspace Manager
- Oracle Multimedia
- Oracle XML Database
- OLAP Analytic Workspace
- Oracle OLAP API
- Oracle Text
- Oracle Spatial and Graph
- Oracle Data Mining
- Oracle Label Security
- Messaging Gateway
- Oracle Database Vault

About Transporting and Upgrading a Database (Full Transportable Export/Import)

You can use file-based or nonfile-based modes for transporting data.

The Full Transportable Export/Import feature of Oracle Data Pump provides two options.

- Using a file-based Oracle Data Pump export/import
- Using a nonfile-based network mode Oracle Data Pump import

> **See Also:**
>
> - *Oracle Database Administrator's Guide* for information about transporting a database using an export dump file
> - *Oracle Database Administrator's Guide* for the procedure to transport a database over the network

Upgrade Scenarios for Non-CDB Oracle Databases

Review these topics to understand the upgrade scenarios and procedures for non-CDB Oracle Databases

> **Note:**
>
> Starting with Oracle Database 12*c*, release 1 (12.1), non-CDB architecture is deprecated. It can be desupported in a future release. Oracle Database deployed with the multitenant architecture is the default configuration option. All Oracle Database releases earlier than Oracle Database 12*c* release 1 (12.1.0.1) use non-CDB architecture.

> **Caution:**
>
> You cannot downgrade a database after you have set the compatible initialization parameter to 12.1.0.2. A pluggable database (PDB) downgrade is possible only if the compatibility is set to 12.1.0.1. There can be additional restrictions on downgrading.
>
> Before starting an upgrade or a downgrade, Oracle strongly recommends that you upgrade your source and target databases to the most recent Quarterly Release Update (Update), Release Update Revision (Revision), bundle patch, or patch set update (BP or PSU).

- Download the Latest Oracle Database Preupgrade Information Tool
  To ensure that you obtain the most recent updates to checks to help to ensure a smooth upgrade, download the most recent Oracle Database Preupgrade tool.
- About Adopting a Non-CDB as a PDB Using a PDB Plugin
  To manually adopt a non-CDB as a PDB, you generate an XML file that describes a non-CDB, and use the `DBMS_PDB.DESCRIBE` procedure. Afterward, plug in the non-CDB, just as you plug in an unplugged PDB.
- Adopting a Non-CDB as a PDB
  You can adopt (move) a non-CDB into a PDB by using the `DBMS_PDB.DESCRIBE` procedure.
- Oracle Label Security Integration in a Multitenant Environment
  You can use Oracle Label Security in a multitenant environment.
- Manually Upgrading Non-CDB Architecture Oracle Databases
  This procedure provides steps for upgrading non-CDB architecture Oracle Databases.
- Upgrading Oracle Database Using Fleet Patching and Provisioning
  In Oracle Database 12c release 2 (12.2) and later releases, you can use Fleet Patching and Provisioning to upgrade an earlier release Oracle Database.

Download the Latest Oracle Database Preupgrade Information Tool

To ensure that you obtain the most recent updates to checks to help to ensure a smooth upgrade, download the most recent Oracle Database Preupgrade tool.

The Oracle Database Preupgrade Information Tool continues to be developed after the initial software release. While the Preupgrade Information Tool that is packaged with your software can identify many issues to facilitate your upgrade, Oracle recommends that you always download and run the most recent version. Downloading the most

recent version of the tool enables you obtain the benefit of new updates added to respond to any issues that emerge after the initial release. The most recent version of the tool is available from My Oracle Support document ID 884522.1.

Before you download the newer version of the Pre-Upgrade Information Tool, back up the version that came with your software. Then replace the files in your Oracle home with the files that you download from Oracle Support.

Related Topics

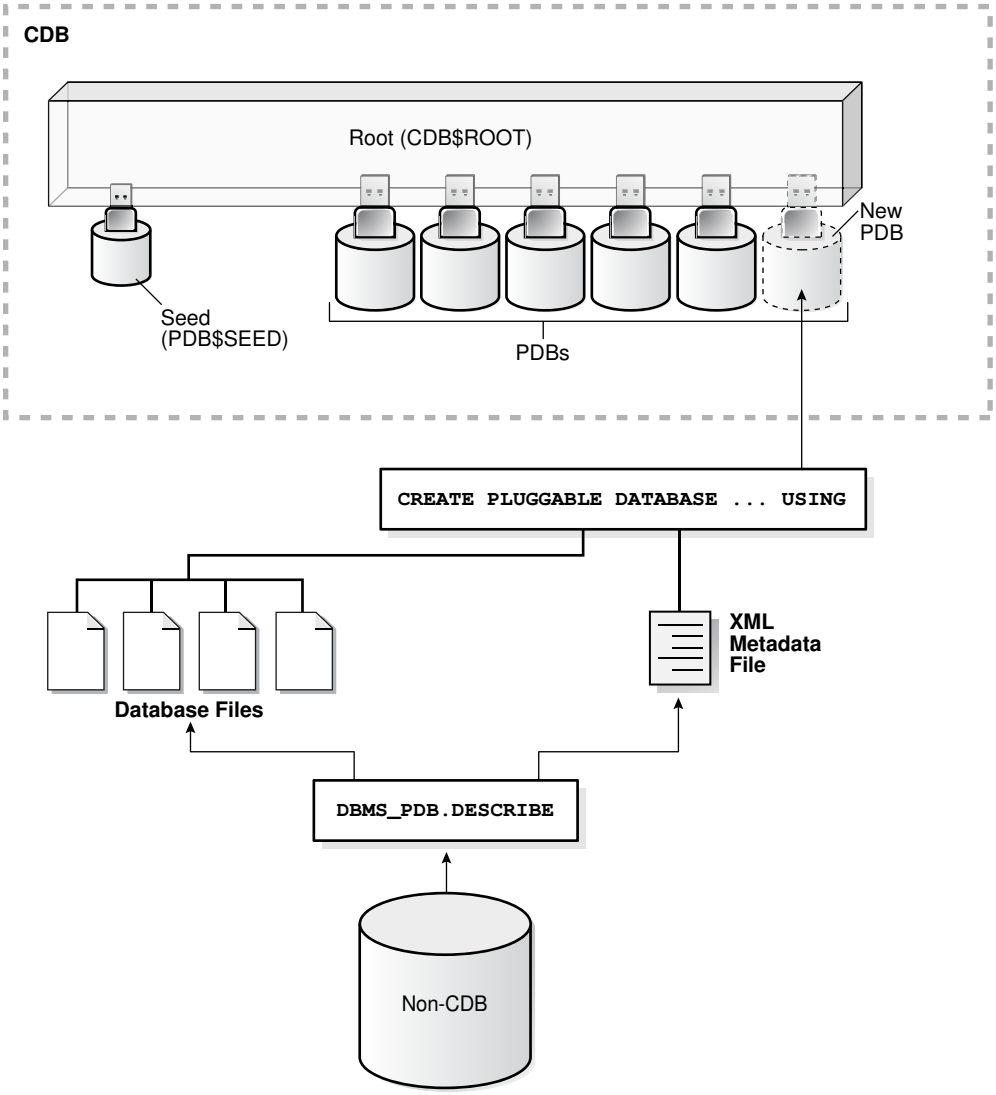
- My Oracle Support Doc ID 884522.1
- Using the Pre-Upgrade Information Tool for Oracle Database
  Review these topics to understand and to use the Pre-Upgrade information tool (`preupgrade.jar`).

About Adopting a Non-CDB as a PDB Using a PDB Plugin

To manually adopt a non-CDB as a PDB, you generate an XML file that describes a non-CDB, and use the `DBMS_PDB.DESCRIBE` procedure. Afterward, plug in the non-CDB, just as you plug in an unplugged PDB.

If you choose not to use the Capture Replay method of automatically adopting and upgrading a non-CDB to a PDB, then you can use the manual procedure of describing the non-CDB, and then adopting the non-CDB to a PDB. Create the PDB with the `CREATE PLUGGABLE DATABASE ... USING` statement. When the non-CDB is plugged in to a CDB, it is a new PDB, but not usable until the data dictionary is converted, using the *`ORACLE_HOME`*`/rdbms/admin/noncdb_to_pdb.sql` script.

Figure 4-1 Plug In a Non-CDB Using the DBMS_PDB.DESCRIBE Procedure



You can use the same technique to create a new application PDB in an application container.

Adopting a Non-CDB as a PDB

You can adopt (move) a non-CDB into a PDB by using the `DBMS_PDB.DESCRIBE` procedure.

This procedure enables you to update your non-CDB Oracle Database to a PDB on a CDB. To use this procedure, you must first install a new Oracle Database release with a CDB.

1. Create the CDB if it does not exist.
2. Ensure that the non-CDB is in a transactionally-consistent state.
3. Place the non-CDB in read-only mode.

4. Connect to the non-CDB, and run the `DBMS_PDB.DESCRIBE` procedure to construct an XML file that describes the non-CDB.

   The current user must have `SYSDBA` administrative privilege. The user must exercise the privilege using `AS SYSDBA` at connect time.

   For example, to generate an XML file named `ncdb.xml` in the `/disk1/oracle` directory, run the following procedure:

   ```
   BEGIN
     DBMS_PDB.DESCRIBE(
       pdb_descr_file => '/disk1/oracle/ncdb.xml');
   END;
   /
   ```

   After the procedure completes successfully, you can use the XML file and the non-CDB database files to plug the non-CDB into a CDB.

5. Run the `DBMS_PDB.CHECK_PLUG_COMPATIBILITY` function to determine whether the non-CDB is compatible with the CDB.

   When you run the function, set the following parameters:

   - `pdb_descr_file` - Set this parameter to the full path to the XML file.
   - `pdb_name` - Specify the name of the new PDB. If this parameter is omitted, then the PDB name in the XML file is used.

   For example, to determine whether a non-CDB described by the `/disk1/oracle/ncdb.xml` file is compatible with the current CDB, run the following PL/SQL block:

   ```
   SET SERVEROUTPUT ON
   DECLARE
     compatible CONSTANT VARCHAR2(3) :=
       CASE DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
              pdb_descr_file => '/disk1/oracle/ncdb.xml',
              pdb_name       => 'NCDB')
       WHEN TRUE THEN 'YES'
       ELSE 'NO'
   END;
   BEGIN
     DBMS_OUTPUT.PUT_LINE(compatible);
   END;
   /
   ```

   If the output is `YES`, then the non-CDB is compatible, and you can continue with the next step. If the output is `NO`, then the non-CDB is not compatible, and you can check the `PDB_PLUG_IN_VIOLATIONS` view to see why it is not compatible. All violations must be corrected before you continue. For example, any version or patch mismatches should be resolved by running an upgrade or the datapatch utility. After correcting the violations, run `DBMS_PDB.CHECK_PLUG_COMPATIBILITY` again to ensure that the non-CDB is compatible with the CDB.

6. Shut down the non-CDB.
7. Plug in the non-CDB.

For example, the following SQL statement plugs in a non-CDB, copies its files to a new location, and includes only the tbs3 user tablespace from the non-CDB:

```
CREATE PLUGGABLE DATABASE ncdb USING '/disk1/oracle/ncdb.xml'
  COPY
  FILE_NAME_CONVERT = ('/disk1/oracle/dbs/', '/disk2/oracle/ncdb/')
  USER_TABLESPACES=('tbs3');
```

If there are no violations, then do not open the new PDB. You will open it in the following step.

The USER_TABLESPACES clause enables you to separate data that was used for multiple tenants in a non-CDB into different PDBs. You can use multiple CREATE PLUGGABLE DATABASE statements with this clause to create other PDBs that include the data from other tablespaces that existed in the non-CDB.

8. Run the *ORACLE_HOME*/rdbms/admin/noncdb_to_pdb.sql script. This script must be run before the PDB can be opened for the first time.

   If the PDB was not a non-CDB, then running the noncdb_to_pdb.sql script is not required. To run the noncdb_to_pdb.sql script, complete the following steps:

   a. Access the PDB.

      The current user must have SYSDBA administrative privilege, and the privilege must be either commonly granted or locally granted in the PDB. The user must exercise the privilege using AS SYSDBA at connect time.

   b. Run the noncdb_to_pdb.sql script:

      ```
      @$ORACLE_HOME/rdbms/admin/noncdb_to_pdb.sql
      ```

   The script opens the PDB, performs changes, and closes the PDB when the changes are complete.

9. Open the new PDB in read/write mode.

   You must open the new PDB in read/write mode for Oracle Database to complete the integration of the new PDB into the CDB. An error is returned if you attempt to open the PDB in read-only mode. After the PDB is opened in read/write mode, its status is NORMAL.

10. Back up the PDB.

    A PDB cannot be recovered unless it is backed up.

> **Note:**
>
> If an error is returned during PDB creation, then the PDB being created can be in an UNUSABLE state. To check the state of a PDB, query the CDB_PDBS or DBA_PDBS view. You can learn more about PDB creation errors by checking the alert log. An unusable PDB can only be dropped. You must drop an unusable PDB before you try to create a PDB with the same name as the unusable PDB can be created.

Oracle Label Security Integration in a Multitenant Environment

You can use Oracle Label Security in a multitenant environment.

In a multitenant environment, pluggable databases (PDBs) can be plugged in and out of a multitenant container database (CDB) or an application container.

Note the following:

- Each PDB has its own Oracle Label Security metadata, such as policies, labels, and user authorizations. The `LBACSYS` schema is a common user schema.
- Before you plug a PDB into a CDB, if the database does not have Oracle Label Security installed, then ensure that you have run the `$ORACLE_HOME/rdbms/admin/catols.sql` script on the database to install the label-based framework, data dictionary, data types, and packages. This script creates the `LBACSYS` account.
- Because Oracle Label Security policies are scoped to individual PDBs, you can create individual policies for each PDB. A policy defined for a PDB can be enforced on the local tables and schema objects contained in the PDB.
- In a single CDB, there can be multiple PDBs, each configured with Oracle Label Security.
- You cannot create Oracle Label Security policies in the CDB root or the application root.
- You cannot enforce a local Oracle Label Security policy on a common CDB object or a common application object.
- You cannot assign Oracle Label Security policy labels and privileges to common users and application common users in a pluggable database.
- You cannot assign Oracle Label Security privileges to common procedures or functions and application common procedures or functions in a pluggable database.
- If you are configuring Oracle Label Security with Oracle Internet Directory, then be aware that the same configuration must be used throughout with all PDBs contained in the CDB. You can determine if your database is configured for Oracle Internet Directory by querying the `DBA_OLS_STATUS` data dictionary view as follows from within any PDB:

  ```
  SELECT STATUS FROM DBA_OLS_STATUS WHERE NAME = 'OLS_DIRECTORY_STATUS';
  ```

  If it returns `TRUE`, then Oracle Label Security is Internet Directory-enabled. Otherwise, it returns `FALSE`.

Related Topics

- *Oracle Database Security Guide*

Manually Upgrading Non-CDB Architecture Oracle Databases

This procedure provides steps for upgrading non-CDB architecture Oracle Databases.

> **Note:**
>
> Starting with Oracle Database 12c Release 1 (12.1), non-CDB architecture is deprecated. It can be desupported in a future release.

Before using this procedure, complete the following steps:

- Install the Oracle Database software
- Prepare the new Oracle home
- Run the Pre-Upgrade Information Tool

Steps:

1. If you have not done so, run the Pre-Upgrade Information Tool. Review the Pre-Upgrade Information Tool output and correct all issues noted in the output before proceeding.

   For example, on Linux or Unix systems:

   ```
   $ORACLE_HOME/jdk/bin/java -jar /opt/oracle/product/19.0.0/rdbms/
   admin/preupgrade.jar FILE TEXT
   ```

2. Ensure that you have a proper backup strategy in place.
3. If you have not done so, prepare the new Oracle home.
4. (Conditional) For Oracle RAC environments only, enter the following commands to set the initialization parameter value for CLUSTER_DATABASE to FALSE:

   ```
   ALTER SYSTEM SET CLUSTER_DATABASE=FALSE SCOPE=SPFILE;
   ```

5. Shut down the database. For example:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

6. If your operating system is Windows, then complete the following steps:

   a. Stop the `OracleService`*`SID`* 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`. Refer to your platform guide for a complete list of the `ORADIM` syntax and commands.

      For example, if your *`SID`* is `ORCL`, then enter the following command.

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

   c. Create the service for the new release Oracle Database at a command prompt using the `ORADIM` command of the new Oracle Database release.

Use the following syntax, where *SID* is your database SID, *PASSWORD* is your system password, *USERS* is the value you want to set for maximum number of users, and *ORACLE_HOME* is your Oracle home:

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

Most Oracle Database services log on to the system using the privileges of the Oracle software installation owner. The service runs with the privileges of this user. The `ORADIM` command prompts you to provide the password to this user account. You can specify other options using `ORADIM`.

In the following example, if your *SID* is `ORCL`, your *password* (`SYSPWD`) is `TWxy5791`, the maximum number of users (`MAXUSERS`) is 10, and the Oracle home path is `C:\ORACLE\PRODUCT\19.0.0\DB`, then enter the following command:

```
C:\> ORADIM -NEW -SID ORCL -SYSPWD TWxy5791 -MAXUSERS 10
-STARTMODE AUTO -PFILE
C:\ORACLE\PRODUCT\19.0.0\DB\DATABASE\INITORCL.ORA
```

`ORADIM` writes a log file to the `ORACLE_HOME\database` directory.

> **Note:**
>
> If you use an Oracle Home User account to own the Oracle home, then the ORADIM command prompts you for that user name and password.

7. If your operating system is Linux or UNIX, then perform the following checks:
   a. Your `ORACLE_SID` is set correctly
   b. The `oratab` file points to the new Oracle home
   c. The following environment variables point to the new Oracle Database directories:
      - `ORACLE_HOME`
      - `PATH`
   d. Any scripts that clients use to set the `$ORACLE_HOME` environment variable must point to the new Oracle home.

> **Note:**
>
> If you are upgrading an Oracle Real Application Clusters database, then perform these checks on all Oracle Grid Infrastructure nodes where the Oracle Real Application Clusters database has instances configured.

8. Log in to the system as the Oracle installation owner for the new Oracle Database release.

9. Start SQL*Plus in the new Oracle home from the admin directory in the new Oracle home directory.

   For example:

   ```
   $ cd $ORACLE_HOME/rdbms/admin
   $ pwd
   /u01/app/oracle/product/19.0.0/dbhome_1/rdbms/admin
   $ sqlplus
   ```

10. Copy the SPFILE.ORA or INIT.ORA file from the old Oracle home to the new Oracle home.
11. Connect to the database that you want to upgrade using an account with SYSDBA privileges:

    ```
    SQL> connect / as sysdba
    ```

12. Start the non-CDB Oracle Database in upgrade mode:

    ```
    SQL> startup upgrade
    ```

    If errors appear listing desupported initialization parameters, then make a note of the desupported initialization parameters and continue with the upgrade. Remove the desupported initialization parameters the next time you shut down the database.

    > **Note:**
    >
    > Starting up the database in `UPGRADE` mode enables you to open a database based on an earlier Oracle Database release. It also restricts log-ins to `AS SYSDBA` sessions, disables system triggers, and performs additional operations that prepare the environment for the upgrade.

13. Exit SQL*Plus.

    For example:

    ```
    SQL> EXIT
    ```

14. Run the Parallel Upgrade Utility (`catctl.pl`) script, using the upgrade options that you require for your upgrade.

    You can run the Parallel Upgrade Utility as a command-line shell command by using the `dbupgrade` shell command, which is located in *`Oracle_home`*`/bin`. If you set the PATH environment variable to include `Oracle_home/bin`, then you can run the command directly from your command line. For example:

    ```
    $ dbupgrade
    ```

> **Note:**
>
> - When you run the Parallel Upgrade Utility command, you can use the `-l` option to specify the directory that you want to use for spool log files.

15. The database is shut down after a successful upgrade. Restart the instance so that you reinitialize the system parameters for normal operation. For example:

    ```
    SQL> STARTUP
    ```

    This restart, following the database shutdown, flushes all caches, clears buffers, and performs other housekeeping activities. These measures are an important final step to ensure the integrity and consistency of the upgraded Oracle Database software.

    > **Note:**
    >
    > If you encountered a message listing desupported initialization parameters when you started the database, then remove the desupported initialization parameters from the parameter file before restarting it. If necessary, convert the `SPFILE` to a `PFILE`, so that you can edit the file to delete parameters.

16. Run `catcon.pl` to start `utlrp.sql`, and to recompile any remaining invalid objects.

    For example:

    ```
    $ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlrp -d '''.''' 
    utlrp.sql
    ```

    Because you run the command using `-b utlrp`, the log file `utlrp0.log` is generated as the script is run. The log file provides results of the recompile.

17. Run `postupgrade_fixups.sql`. For example:

    ```
    SQL> @postupgrade_fixups.sql
    ```

    > **Note:**
    >
    > If you did not specify to place the script in a different location, then it is in the default path *`Oracle_base`*`/cfgtoollogs/`*`SID`*`/preupgrade`, where *`Oracle_base`* is your Oracle base home path, and *`SID`* is your unique database name.

18. Run `utlusts.sql`. The script verifies that all issues are fixed.

For example:

```
SQL> @$ORACLE_HOME/rdbms/admin/utlusts.sql
```

The log file `utlrp0.log` is generated as the script is run, which provides the upgrade results. You can also review the upgrade report in `upg_summary.log`.

To see information about the state of the database, run `utlusts.sql` as many times as you want, at any time after the upgrade is completed. If the `utlusts.sql` script returns errors, or shows components that do not have the status `VALID`, or if the version listed for the component is not the most recent release, then refer to the troubleshooting section in this guide.

19. Ensure that the time zone data files are current by using the `DBMS_DST PL/SQL` package to upgrade the time zone file. You can also adjust the time zone data files after the upgrade.
20. Exit from SQL*Plus

    For example:

    ```
    SQL> EXIT
    ```

21. (Conditional) If you are upgrading an Oracle Real Application Clusters database, then use the following command syntax to upgrade the database configuration in Oracle Clusterware:

    ```
    srvctl upgrade database -db db-unique-name -oraclehome oraclehome
    ```

    In this syntax example, *`db-unique-name`* is the database name (not the instance name), and *`oraclehome`* is the Oracle home location in which the database is being upgraded. The SRVCTL utility supports long GNU-style options, in addition to short command-line interface (CLI) options used in earlier releases.
22. (Conditional) For Oracle RAC environments only, after you have upgraded all nodes, enter the following commands to set the initialization parameter value for CLUSTER_DATABASE to TRUE, and start the database, where *`db_unique_name`* is the name of the Oracle RAC database:

    ```
    ALTER SYSTEM SET CLUSTER_DATABASE=TRUE SCOPE=SPFILE;
    srvctl start database -db db_unique_name
    ```

Your database is now upgraded. You are ready to complete post-upgrade procedures.

> **Caution:**
>
> If you retain the old Oracle software, then never start the upgraded database with the old software. Only start Oracle Database using the start command in the new Oracle Database home.
>
> Before you remove the old Oracle environment, relocate any data files in that environment to the new Oracle Database environment.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for information about relocating data files

Upgrading Oracle Database Using Fleet Patching and Provisioning

In Oracle Database 12c release 2 (12.2) and later releases, you can use Fleet Patching and Provisioning to upgrade an earlier release Oracle Database.

You upgrade a database with Fleet Patching and Provisioning by creating a copy of the new Oracle Database release, and using the command `rhpctl upgrade database` to upgrade the earlier release Oracle Database. The upgrade is an out-of-place upgrade. After the upgrade is complete, listeners and other initialization variables are set to point to the new Oracle home. Refer to *Oracle Clusterware Administration and Deployment Guide* for more information about how to create Fleet Patching and Provisioning images.

Use this overview of the steps to understand how to upgrade an earlier Oracle Database release by using Fleet Patching and Provisioning:

1. Install a new Oracle Database release.
2. Patch, test, and configure the database to your specifications for a standard operating environment (SOE).
3. Create a Fleet Patching and Provisioning Gold Image from the SOE release Oracle Database home.
4. Complete an upgrade to a new Oracle Grid Infrastructure release on the servers where the databases you want to upgrade are located. You can complete this upgrade by using Fleet Patching and Provisioning. (Note: Your Oracle Grid Infrastraucture software must always be the same or a more recent release than Oracle Database software.)
5. Deploy a copy of the new release Oracle Database Fleet Patching and Provisioning gold image to the servers with earlier release Oracle Databases that you want to upgrade.
6. Run the Fleet Patching and Provisioning command `rhpctl upgrade database`. This command use the new release Fleet Patching and Provisioning gold image to upgrade the earlier release databases. You can upgrade one, many or all of the earlier release Oracle Database instances on the servers provisioned with the new release Oracle Database gold image.

Related Topics

- Fleet Patching and Provisioning

5

Troubleshooting the Upgrade for Oracle Database

Use these troubleshooting tips to address errors or issues that you may encounter while upgrading your database.

Also review the related links to learn about changes that affect this release, which may be related to errors you receive, and to see how to rerun the upgrade after you resolve errors.

- About Starting Oracle Database in Upgrade Mode
  When you start Oracle Database in upgrade mode, you can only run queries on fixed views. If you attempt to run other views or PL/SQL, then you receive errors.
- Running DBUA with Different ORACLE_HOME Owner
  Review this topic if your Oracle Database homes are owned by different operating system user accounts, or you encounter an `upgrade.xml not found` error.
- Invalid Object Warnings and DBA Registry Errors
  Before you start your upgrade, Oracle strongly recommends that you run the preupgrade information tool (`preupgrd.jar`).
- Invalid Objects and Premature Use of Postupgrade Tool
  Never run the postupgrade status tool for the new Oracle Database release (`utlusts.sql`) until after you complete the upgrade.
- Resolving Oracle Database Upgrade Script Termination Errors
  Review this section if you encounter ORA-00942, ORA-00904, or ORA-01722 errors.
- Troubleshooting Causes of Resource Limits Errors while Upgrading Oracle Database
  Review this section if you encounter `ORA-01650`, `ORA-01651`, `ORA-01652`, `ORA-01653`, `ORA-01654`, `ORA-01655`, `ORA-0431`, `ORA-01562`, `ORA-19815`, or other errors that suggest resource limit errors.
- Resolving SQL*Plus Edition Session Startup Error for Oracle Database
  Use this section to understand and resolve SP2–1540: "Oracle Database cannot startup in an Edition session."
- Error ORA-00020 Maximum Number of Processes Exceeded When Running utlrp.sql
  This error may indicate that your Oracle configuration does not have sufficient number of processes available for the recompile.
- Fixing ORA-28365: Wallet Is Not Open Error
  If you use Oracle wallet with Transparent Data Encryption (TDE), and you use Database Upgrade Assistant (DBUA) to upgrade the database, then you can encounter an ORA-28365 "wallet is not open" error.
- Resolving issues with view CDB_JAVA_POLICY
  If the view CDB_JAVA_POLICY becomes invalid, then use this procedure.

- Continuing Upgrades After Server Restarts (ADVM/ACFS Driver Error)
  On Windows platforms, an error may occur related to ADVM or ACFS drivers if a server restarts during an upgrade.
- Component Status and Upgrades
  Component status settings are affected both by the components that you previously installed, and by the support of those components for upgrades.
- Standard Edition Starter Database and Components with Status OPTION OFF
  Starting in Oracle Database 18c (18.1), all `OPTION OFF` components are upgraded to the new release, but these options are disabled for Oracle Database Standard Edition (SE) remain OPTION OFF.
- Adjusting Oracle ASM Password File Location After Upgrade
  You must create a new password file for Oracle ASM after an Oracle Grid Infrastructure upgrade.
- Fixing "Warning XDB Now Invalid" Errors with Pluggable Database Upgrades
  Review this topic if you encounter "Warning: XDB now invalid, invalid objects found" errors when upgrading pluggable databases (PDBs).
- Fixing ORA-27248: sys.dra_reevaluate_open_failures is running
  Use this procedure to identify `DRA_REEVALUATE_OPEN_FAILURES` jobs that block upgrades.
- Fixing Failed Upgrades Where Only Datapatch Fails
  If only datapatch fails during an upgrade, then rerun datapatch directly.

Related Topics

- Rerunning Upgrades for Oracle Database
  Use these options to rerun upgrades.

About Starting Oracle Database in Upgrade Mode

When you start Oracle Database in upgrade mode, you can only run queries on fixed views. If you attempt to run other views or PL/SQL, then you receive errors.

When the database is started in upgrade mode, only queries on fixed views execute without errors. This restriction applies until you either run the Parallel Upgrade Utility (`catctl.pl`) directly, or indirectly by using the `dbupgrade` script). Before running an upgrade script, using PL/SQL on any other view, or running queries on any other view returns an error. If you receive any of the errors described in this section, then issue the `SHUTDOWN ABORT` command to shut down the database, and then correct the problem.

The following list of errors can occur when you attempt to start the new Oracle Database release. Some of these errors write to the alert log, and not to your session.

- `ORA-00401: the value for parameter compatible is not supported by this release`

  The `COMPATIBLE` initialization parameter is set to a value less than `11.2.0`.
- `ORA-39701: database must be mounted EXCLUSIVE for UPGRADE or DOWNGRADE`

  The `CLUSTER_DATABASE` initialization parameter is set to `TRUE` instead of `FALSE`.
- `ORA-39700: database must be opened with UPGRADE option`

  The `STARTUP` command was issued without the `UPGRADE` keyword.

- `Ora-00704: bootstrap failure`

  The path variable can be pointing to the earlier release Oracle home.

- `ORA-00336: log file size xxxx blocks is less than minimum 8192 blocks`

  A redo log file size is less than 4 MB.

If errors appear listing desupported initialization parameters, then make a note of the desupported initialization parameters, and continue with the upgrade. Remove the desupported initialization parameters the next time you shut down the database.

Related Topics

- https://support.oracle.com/rs?type=doc&id=1349722.1

Running DBUA with Different ORACLE_HOME Owner

Review this topic if your Oracle Database homes are owned by different operating system user accounts, or you encounter an `upgrade.xml not found` error.

DBUA upgrades by default assume that both the source Oracle home and the target Oracle home are owned by the same user. If each Oracle home is not owned by the same user, then you must change to database file permissions and pass additional parameters to DBUA. If you do not do this, then during upgrade, the DBUA Prerequisite Checks page reports `upgrade.xml not found` errors. You are not permitted to proceed with the upgrade until this error is corrected.

- All Oracle Database installation owners should have the group that you designate as the `OINSTALL` group (or Oracle Inventory group) as their primary group. Ensure all database files (data files, the redo file, control files, archive log destination, recovery area, SPFILE, and password file) are readable and writable by both the new target release and the source (earlier) release binary owners. If this is not the case, then confirm that each installation owner has the same group as their primary group, and ensure that members of the `OINSTALL` group have read/write access to all of the earlier release and later release Oracle Database files and directories.
- Run DBUA by specifying the `-logdir` command line option, and provide a directory to which both the new release and earlier release binary owners can write. For example: `/tmp`. DBUA uses the directory you designate with the `logdir` parameter to store the output from the Pre-upgrade Information Tool, and to store any DBUA log files generated during the upgrade. You run the Pre-Upgrade Information tool from the earlier release Oracle Database instance as the earlier release Oracle Database installation owner user account.

  For example:

  ```
  dbua -logdir /tmp
  ```

Invalid Object Warnings and DBA Registry Errors

Before you start your upgrade, Oracle strongly recommends that you run the preupgrade information tool (`preupgrd.jar`).

The preupgrade information tool identifies invalid SYS and SYSTEM objects, as well as other invalid objects. Use `utlrp.sql` to recompile invalid objects. If you fail to do

this before an upgrade, then it becomes difficult to determine which objects in your system were invalid before starting the upgrade, and which objects become invalid as a result of the upgrade.

Related Topics

- Using the Pre-Upgrade Information Tool for Oracle Database

Invalid Objects and Premature Use of Postupgrade Tool

Never run the postupgrade status tool for the new Oracle Database release (`utlusts.sql`) until after you complete the upgrade.

Oracle recommends that you run the postupgrade status tool only after the upgrade process is complete, and after you have run `utlrp.sql`. If the postupgrade status tool is run before you run `@utlrp.sql`, then the output of tool may not display the accurate final component status value. If the tool is run before running `utlrp.sql`, then the component status values may not properly reflect the final state. You can only determine the final component state after running `utlrp.sql`.

Resolving Oracle Database Upgrade Script Termination Errors

Review this section if you encounter ORA-00942, ORA-00904, or ORA-01722 errors.

If you do not run the Pre-Upgrade Information Tool before starting the upgrade, then the `catctl.pl` and `catupgrd.sql` scripts terminate with errors as follows:

```
ORA-00942: table or view does not exist
ORA-00904: "TZ_VERSION": invalid identifier
ORA-01722: invalid number
```

If you receive any of these errors, then complete the following procedure:

1. Enter a `SHUTDOWN ABORT` command, and wait for the command to complete running.
2. Revert to the original Oracle home directory
3. Run the Pre-Upgrade Information Tool.

Related Topics

- Pre-Upgrade Information Tool for Oracle Database
  Review these topics to understand and to use the Pre-Upgrade information tool (`preupgrade.jar`).

Troubleshooting Causes of Resource Limits Errors while Upgrading Oracle Database

Review this section if you encounter `ORA-01650`, `ORA-01651`, `ORA-01652`, `ORA-01653`, `ORA-01654`, `ORA-01655`, `ORA-0431`, `ORA-01562`, `ORA-19815`, or other errors that suggest resource limit errors.

If you run out of resources during an upgrade, then increase the resource allocation. After increasing the resource allocation, shut down the instance with `SHUTDOWN ABORT`, and restart the instance in `UPGRADE` mode before re-running the `catupgrd.sql` script. If you run the upgrade using the Parallel Upgrade Utility (`catctl.pl`), then to automatically resume the upgrade from the point of failure, run the utility with the `-R` option. After you fix issues, AutoUpgrade automatically resumes upgrades.

The resources that generally require increases for a new Oracle Database release are as follows:

- `SYSTEM` and `SYSAUX` tablespaces

  If your `SYSTEM` tablespace size is insufficient, then typically you receive the following error message:

  ```
  ORA-01650: unable to extend rollback segment string by string in
  tablespace string
  ORA-01651: unable to extend save undo segment by string for
  tablespace string
  ORA-01652: unable to extend temp segment by string in tablespace
  string
  ORA-01653: unable to extend table string.string by string in
  tablespace string
  ORA-01654: unable to extend index string.string by string in
  tablespace string
  ORA-01655: unable to extend cluster string.string by string in
  tablespace string
  ```

  To avoid these errors, set `AUTOEXTEND ON MAXSIZE UNLIMITED` for the `SYSTEM` and `SYSAUX` tablespaces.

- Shared memory

  In some cases, you may require larger shared memory pool sizes. The error message indicates which shared memory initialization parameter you must increase, in the following format:

  ```
  ORA-04031: unable to allocate string bytes of shared memory
  ("string","string","string","string")
  ```

  > **See Also:**
  >
  > *Oracle Database Administrator's Guide* for information about using manual shared memory management

- Rollback segments/undo tablespace

  If you are using rollback segments, then you must have a single large (100 MB) `PUBLIC` rollback segment online while the upgrade scripts are being run. Smaller public rollback segments should be taken offline during the upgrade. If your

rollback segment size is insufficient, then typically you encounter the following error:

```
ORA-01562: failed to extend rollback segment number string
```

If you are using an undo tablespace, then be sure it is at least 400 MB.

- Fast Recovery Area

  If you are using a Fast Recovery Area and it fills up during the upgrade, then the following error appears in the alert log, followed by suggestions for recovering from the problem:

  ```
  ORA-19815: WARNING: db_recovery_file_dest_size of string bytes is
  98.99%
  used, and has string remaining bytes available.
  ```

  Identify the root cause of the problem, and take appropriate actions to proceed with the upgrade. To avoid issues during the upgrade, increase the amount of space available in your Fast Recovery Area before starting the upgrade.

Resolving SQL*Plus Edition Session Startup Error for Oracle Database

Use this section to understand and resolve SP2–1540: "Oracle Database cannot startup in an Edition session."

If an upgrade script or a command running in SQL*Plus set the `EDITION` parameter, then Oracle Database cannot start properly afterward. When you attempt to start the database, you receive the following error:

```
SP2-1540: "Oracle Database cannot startup in an Edition session"
```

To avoid this problem, after running `catugrd.sql` or any SQL*Plus session where this parameter is changed, exit the SQL*Plus session and restart the instance in a different session.

Error ORA-00020 Maximum Number of Processes Exceeded When Running utlrp.sql

This error may indicate that your Oracle configuration does not have sufficient number of processes available for the recompile.

Refer to Oracle documentation for more details about setting the `PROCESSES` parameter.

> **See Also:**
>
> *Oracle Database Reference*

Fixing ORA-28365: Wallet Is Not Open Error

If you use Oracle wallet with Transparent Data Encryption (TDE), and you use Database Upgrade Assistant (DBUA) to upgrade the database, then you can encounter an ORA-28365 "wallet is not open" error.

To avoid this problem, complete the following tasks before upgrading:

1. Log in as an authorized user.
2. Manually copy the `sqlnet.ora` file, and the wallet file, `ewallet.p12`, to the new release Oracle home.
3. Open the Oracle wallet in mount.

   For example:

   ```
   SQL> STARTUP MOUNT;
   SQL> ALTER SYSTEM SET ENCRYPTION WALLET OPEN
   ```

4. Start the upgrade as usual.

Resolving issues with view CDB_JAVA_POLICY

If the view CDB_JAVA_POLICY becomes invalid, then use this procedure.

After an upgrade to Oracle Database 12*c* release 2 (12.2) and later releases, or a downgrade from release 12.2 or later releases to 12.1, you can encounter issues with the CDB_JAVA_POLICY view. CDB_JAVA_POLICY can become invalid, or it can encounter errors when you use it in a manner that normally works. If this happens, then connect as SYS, and run the following commands.

Non-CDBs:

```
alter session set "_ORACLE_SCRIPT"=true;

exec
CDBView.create_cdbview(false,'SYS','dba_java_policy','CDB_java_policy');

grant select on SYS.CDB_java_policy to select_catalog_role
/
create or replace public synonym CDB_java_policy for SYS.CDB_java_policy
/
```

Multitenant architecture systems:

Run these same commands, but run them first in CDB$ROOT, and then in other containers in the CDB.

Continuing Upgrades After Server Restarts (ADVM/ACFS Driver Error)

On Windows platforms, an error may occur related to ADVM or ACFS drivers if a server restarts during an upgrade.

If a server restarts during the upgrade, then you may see one of the following error messages:

```
ACFS-9427: Failed to unload ADVM/ACFS drivers. A system reboot is
recommended


ACFS-9428 Failed to load ADVM/ACFS drivers. A system reboot is
recommended.
```

- Cause

  The ADVM and ACFS drivers are still in use. You must restart the system to start the new drivers.

- Action

  Complete the steps as described in the following procedures.

For nodes other than the first node (the node on which the upgrade is started):

1. Restart the node where the error occurs.
2. Run the root script on that node again.

For first nodes (the node on which the upgrade is started):

1. Complete the upgrade of all other nodes in the cluster.
2. Restart the first node.
3. Run the root script on the first node again.
4. To complete the upgrade, log in as root, and run the script `configToolAllCommands`, located in the path *`Grid_home`*`/cfgtoollogs/configToolAllCommands`.

> **See Also:**
>
> *Oracle Grid Infrastructure Installation Guide* for your operating system for more information about troubleshooting upgrade issues for clusters

Component Status and Upgrades

Component status settings are affected both by the components that you previously installed, and by the support of those components for upgrades.

Topics:

- Understanding Component Status With the Post-Upgrade Status Tool
  The Post-Upgrade Status tool, `utlusts.sql`, reports database component status after an upgrade is completed.
- Component OPTION OFF Status and Upgrades
  The upgrade status of `OPTION OFF` components is affected both by the support in the target release for a component, and if a component must be upgraded as part of an upgrade.
- Example of an Upgrade Summary Report
  Upgrade summary reports provide information about the upgrade status of components.

Understanding Component Status With the Post-Upgrade Status Tool

The Post-Upgrade Status tool, `utlusts.sql`, reports database component status after an upgrade is completed.

You can run the Post-Upgrade Status Tool `utlusts.sql` anytime after upgrade, post-upgrade, or after recompiling invalid objects with `utlrp.sql`.

The following list briefly describes the status values that the Post-Upgrade Status tool reports:

- `INVALID`

  When the upgrade completed, some objects for the component remained in an invalid state. If you find no errors in the log file for component upgrades then run the script `utlrp.sql`. Running this script may change the status of invalid components to `VALID` without rerunning the entire upgrade. Check the `DBA_REGISTRY` view after running `utlrp.sql`.
- `VALID`

  The component is valid with no errors.
- `LOADING`

  The component is loading
- `LOADED`

  The component has successfully finished loading.
- `UPGRADING`

  The component is in process being upgraded.
- `UPGRADED`

  The component has completed upgrading with no errors.
- `DOWNGRADING`

  The component is in process being downgraded.
- `DOWNGRADED`

  The component has completed downgrading with no errors.
- `REMOVING`

  The component is in process being removed.

- REMOVED

  The component was not upgraded because it was removed from the database.

- OPTION OFF

  The server option required for the component was not installed or was not linked with the server. Check the V$OPTION view and the install logs. Install the component or relink the server with the required option, and then re-run the Parallel Upgrade Utility.

- NO SCRIPT

  The component upgrade script was not found in $ORACLE_HOME. Check the install logs, install the component software, and then re-run the Parallel Upgrade Utility.

> **Note:**
>
> You can run the Parallel Upgrade Utility (catctl.pl using the command-line scripts dbupgrade, on all supported platforms).

Component OPTION OFF Status and Upgrades

The upgrade status of OPTION OFF components is affected both by the support in the target release for a component, and if a component must be upgraded as part of an upgrade.

There are three cases where OPTION OFF components are upgraded, or are not upgraded.

Unsupported Components With Status OPTION OFF

If there is a component in the database that is in the status OPTION OFF, and that component is no longer supported for database upgrades to the target release, then this component is not upgraded. After the upgrade, its version and status remain unchanged.

Supported Components With Status OPTION OFF

If there is a component in the database that is in the status OPTION OFF, but that component is supported for database upgrades to the target release, then this component is upgraded. After the upgrade, the component's version matches the target release version. The status for this component is either UPGRADED (a successful upgrade), or INVALID (errors). Rerun the upgrade as needed, until all the upgraded components have a status of UPGRADED. Then run utlrp.sql. If a component was in the status OPTION OFF before the upgrade, then after it is upgraded, and its compile and validation is successful, its status reverts back to OPTION OFF.

Supported Components With Required Options That Must Be Upgraded

All components with required options must be upgraded. These components are:

- RAC
- SDO
- APS

- XOQ

Components that must be upgraded follow the same procedure for upgrades as for standard supported components with status OPTION OFF

Example of an Upgrade Summary Report

Upgrade summary reports provide information about the upgrade status of components.

After the upgrade completes, starting with Oracle Database 12*c* release 2, the upgrade utility script (for example, in Oracle Database 19c, utlusts.sql) writes a copy of the report using the following write order:

1. ORACLE_BASE/cfgtoollogs/*db_unique_name*/upgrade*YYMMDDHHMMSC*/upg_summary.rpt
2. $ORACLE_HOME/rdbms/log/upg_summary.rpt
3. Linux and UNIX:

   /tmp/upg_summary.rpt

   Windows:

   \TEMP\upg_summary.rpt

If the utility script cannot create the report under the first directory scheme, then it attempts to write to the second location, and then the third location. If it cannot write to any of these paths, then it does not write an upgrade summary report. This example is from Oracle Database 18c.

Example 5-1 Upgrade Summary Report for the Post-Upgrade Status Tool

```
Oracle Database Release 18 Post-Upgrade Status Tool    05-27-2018
23:41:0

Component                               Current         Full
Elapsed Time
Name                                    Status          Version
HH:MM:SS

Oracle Server                          UPGRADED      18.3.0.0.0
00:18:21
JServer JAVA Virtual Machine           UPGRADED      18.3.0.0.0
00:04:31
Oracle XDK                             UPGRADED      18.3.0.0.0
00:00:51
Oracle Database Java Packages          UPGRADED      18.3.0.0.0
00:00:22
OLAP Analytic Workspace                UPGRADED      18.3.0.0.0
00:00:51
Oracle Label Security                  UPGRADED      18.3.0.0.0
00:00:20
Oracle Database Vault                  UPGRADED      18.3.0.0.0
```

```
00:00:40
Oracle Text                              UPGRADED      18.3.0.0.0
00:01:37
Oracle Workspace Manager                 UPGRADED      18.3.0.0.0
00:01:10
Oracle Real Application Clusters         UPGRADED      18.3.0.0.0
00:00:00
Oracle XML Database                      UPGRADED      18.3.0.0.0
00:02:17
Oracle Multimedia                        UPGRADED      18.3.0.0.0
00:03:01
Spatial                                  UPGRADED      18.3.0.0.0
00:06:23
Oracle OLAP API                          UPGRADED      18.3.0.0.0
00:00:22
Upgrade Datapatch
00:00:10
Final Actions
00:00:20
Post Upgrade
00:00:05
Post Upgrade Datapatch
00:00:04

Total Upgrade Time: 00:41:45
```

Standard Edition Starter Database and Components with Status OPTION OFF

Starting in Oracle Database 18c (18.1), all `OPTION OFF` components are upgraded to the new release, but these options are disabled for Oracle Database Standard Edition (SE) remain OPTION OFF.

When you upgrade Oracle Database Standard Edition (SE) starter databases, the components that are not included with starter databases are turned on and upgraded. When `utlrp.sql` is run, options that are not turned on with your server and not included with SE are reset to `OPTION OFF` in the `DBA_REGISTRY` view.

Adjusting Oracle ASM Password File Location After Upgrade

You must create a new password file for Oracle ASM after an Oracle Grid Infrastructure upgrade.

The Oracle ASM password file location is not shown in the command output when you run `srvctl config asm` after a Grid Infrastructure upgrade. The location of the password file is not automatically passed to the new Oracle ASM disk group. To enable SRVCTL to have the password file location after upgrade, you must advance the diskgroup compatibility setting and create a PWFILE in the disk group. Then SRVCTL reports the configured location of the shared PWFILE.

> **See Also:**
>
> *Oracle Automatic Storage Management Administrator's Guide* for information about managing shared password files in disk groups

Fixing "Warning XDB Now Invalid" Errors with Pluggable Database Upgrades

Review this topic if you encounter "Warning: XDB now invalid, invalid objects found" errors when upgrading pluggable databases (PDBs).

You can encounter XML object errors when you plug an Oracle Database 12*c* release 1 (12.1) pluggable database (PDB) into an Oracle Database 12*c* release 2 (12.2) or later release multitenant container database (CDB).

Common objects (objects with `sharing='METADATA LINK'` in `dba_objects`) are created by registering system-generated names in an object-relational XML schema. Those common types are created by registering some ORDSYS schemas with object-relational storage.

The names of these common objects are system-generated, and the names generated in release 12.1 can be different from the names used for these objects in release 12.2 and later releases. Because of these possible name changes, you can find that the release 12.1 object types do not have matching common types in the release 12.2 or later release CDB root.

Resolve this issue using the following procedure:

1. Query the view `PDB_PLUG_IN_VIOLATIONS` in the target CDB root to see if there is any action containing `'GetTypeDDL'`

   If you find `'GetTypeDDL'` actions, then the upgraded PDB has the common objects issue.

2. Run the PL/SQL packages `SET SERVEROUTPUT ON` and `exec xdb.DBMS_XMLSTORAGE_MANAGE.GetTypeDDL` in the target PDB to generate a user-named SQL script (for example, `script1.sql`).

3. Run the script you created in step 2 (for example, `script1.sql` in the source 12.1 CDB to obtain the type creation script for each of the common types for which you are encountering errors

4. Generate another user-named SQL script (for example, `script2.sql`) that contains these creation scripts.

5. Run the script that you created on the source 12.1 CDB (for example, `script2.sql`) in the target PDB.

The script that you generate from the release 12.1 source CDB type creation scripts generates all of these objects in the target PDB. Making these common objects available in the target PDB should eliminate the invalid XDB errors.

Fixing ORA-27248: sys.dra_reevaluate_open_failures is running

Use this procedure to identify DRA_REEVALUATE_OPEN_FAILURES jobs that block upgrades.

During an upgrade, if DBUA fails with the error ORA-27248: sys.dra_reevaluate_open_failures is running, then the job DRA_REEVALUATE_OPEN_FAILURES is running, which causes upgrade failures. Ensure that the job is stopped before continuing the upgrade.

In a job definition, if ALLOW_RUNS_IN_RESTRICTED_MODE is set to TRUE, and the job owner is permitted to log in during restricted mode, then that job is permitted to run when the database is in restricted or upgrade mode. The default setting for this parameter is FALSE.

Use the following query to see the state of any running jobs:

```
SQL> select OBJECT_NAME, Owner, OBJECT_TYPE from dba_objects whereobject_name
like '%DRA_REEVA%';
```

Fixing Failed Upgrades Where Only Datapatch Fails

If only datapatch fails during an upgrade, then rerun datapatch directly.

The Datapatch script is a shell script. In some patching operations, the final post-upgrade patches may not run, due to errors such as ORA-20001. If only the Datapatch script fails, then you do not need to run the upgrade again to fix this issue. Instead, run the datapatch script directly.

To fix a failed datapatch, log in as the Oracle user, and complete this procedure:

1. Change directory to Opatch inside the upgraded Oracle home.

   ```
   $ cd $ORACLE_HOME/OPatch
   ```

2. Run the datapatch command.

   On Linux and Unix:

   ```
   ./datapatch -verbose
   ```

   On Microsoft Windows:

   ```
   datapatch -verbose
   ```

6

Post-Upgrade Tasks for Oracle Database

After you upgrade Oracle Database, complete required post-upgrade tasks, and consider recommendations for the new release.

- Check the Upgrade With Post-Upgrade Status Tool
  Review the upgrade spool log file and use the Post-Upgrade Status Tool, `utlusts.sql`.
- How to Show the Current State of the Oracle Data Dictionary
  Use one of three methods to check the state of the Oracle Data Dictionary for diagnosing upgrades and migrations.
- Required Tasks to Complete After Upgrading Oracle Database
  Review and complete these required tasks that are specified for your environment after you complete your upgrade.
- Recommended and Best Practices to Complete After Upgrading Oracle Database
  Oracle recommends that you complete these good practices guidelines for updating Oracle Database. Except where noted, these practices are recommended for all type of upgrades (Manual, DBUA or AutoUpgrade).
- Recommended Tasks After Upgrading an Oracle RAC Database
  Decide if you want to configure clients to use SCAN or node listeners for connections.
- Recommended Tasks After Upgrading Oracle ASM
  After you have upgraded Oracle ASM, Oracle recommends that you perform tasks such as resetting the Oracle ASM passwords and configuring disk groups.
- Recommended Tasks After Upgrading Oracle Database Express Edition
  Use DBCA or run manual scripts to install additional components into Oracle Database.
- Tasks to Complete Only After Manually Upgrading Oracle Database
  After you complete your upgrade, you must perform the tasks described here if you upgrade your database manually instead of using DBUA.
- Release Updates and Requirements for Upgrading Oracle Database
  Before starting upgrades, update your new release Oracle home to the latest Release Update (Update).

Release Updates and Requirements for Upgrading Oracle Database

Before starting upgrades, update your new release Oracle home to the latest Release Update (Update).

The software for new Oracle Database releases contains a full release that includes all the latest updates for Oracle Database at the time of the release.

Before you start an upgrade, Oracle strongly recommends that you update your new release Oracle home to the latest quarterly Release Update (Update).

My Oracle Support provides detailed notes about how you can obtain the updates, as well as tools for lifecycle management.. For example:

- My Oracle Support note 2118136.2 contains a download assistant to help you select the updates, revisions, Patch Set Updates (PSU), SPU (CPU), Bundle Patches, Patchsets, and Base Releases that you need for your environment. Oracle highly recommends that you start here.
- My Oracle Support note 1227443.1 contains a list of Oracle Database PSU/BP/Update/Revision known issues. This note provides information about all known issues notes for Oracle Database, Oracle Grid Infrastructure, and the Oracle JavaVM Component (OJVM).

Related Topics

- My Oracle Support Note 2118136.2
- My Oracle Support Note 1227443.1

Check the Upgrade With Post-Upgrade Status Tool

Review the upgrade spool log file and use the Post-Upgrade Status Tool, `utlusts.sql`.

The Post-Upgrade Status Tool is a SQL script that is included with Oracle Database. You run the Post-Upgrade Status Tool in the environment of the new release. You can run the Post-Upgrade Status Tool at any time after you upgrade the database.

How to Show the Current State of the Oracle Data Dictionary

Use one of three methods to check the state of the Oracle Data Dictionary for diagnosing upgrades and migrations.

Running the `dbupgdiag.sql` Script

The `dbupgdiag.sql` script collects upgrade and migration diagnostic information about the current state of the data dictionary.

You can run the script in SQL*Plus both before the upgrade on the source database, and after the upgrade on the upgraded database as the SYS user. Refer to My Oracle Support note 556610.1 for more information about using the `dbupgdiag.sql` script to collect upgrade and migrate diagnostic information.

Running a SQL Query on DBA_REGISTERY

To show the current state of the dictionary, perform a SQL query similar to the following example:

```
SQL> spool /tmp/regInvalid.out
SQL> set echo on
-- query registry
SQL> set lines 80 pages 100
```

```
SQL> select substr(comp_id,1,15) comp_id,substr(comp_name,1,30)
      comp_name,substr(version,1,10) version,status
from dba_registry order by modified;
```

Running a Query to Check for Invalid Objects

To query invalid objects, perform a SQL query similar to the following example:

```
SQL> select owner, object_name, object_type from dba_invalid_objects
order by owner, object_type;
```

After you have upgraded the database, and you have run `utlrp.sql`, this view query should return no rows.

Related Topics

- https://support.oracle.com/rs?type=doc&id=556610.1

Required Tasks to Complete After Upgrading Oracle Database

Review and complete these required tasks that are specified for your environment after you complete your upgrade.

You must complete these postupgrade tasks after you upgrade Oracle Database. You must complete these tasks both when you perform the upgrade manually, and when you upgrade by using Database Upgrade Assistant (DBUA).

- Setting Environment Variables on Linux and Unix Systems After Manual Upgrades
  If you performed a manual upgrade of Oracle Database, then you must ensure that the required operating system environment variables point to the directories of the new Oracle Database release.
- Recompiling All Invalid Objects
  To identify and recompile invalid objects on the CDB and PDBs, use the `catcon` utility to run `utlrp.sql` after you install, patch, or upgrade a database.
- Recompiling All Invalid Objects on Multitenant Architecture Databases
  On multitenant architecture Oracle Databases, after upgrading. Oracle recommends that you recompile invalid objects using the `catcon` Perl script.
- Track Invalid Object Recompilation Progress
  Use these SQL queries to track the progress of `utlrp.sql` script recompilation of invalid objects.
- Setting oratab and Scripts to Point to the New Oracle Location After Upgrading Oracle Database
  You must set scripts to point to the new Oracle home location.
- Check PL/SQL Packages and Dependent Procedures
  It is possible that packages that you installed in the earlier release Oracle Database are not available in the new release, which can affect applications.
- Upgrading Tables Dependent on Oracle-Maintained Types
  Starting with Oracle Database 12*c* Release 2 (12.2) and later releases, you must manually upgrade user tables that depend on Oracle-Maintained types.

- Adjusting Minimum and Maximum for Parallel Execution Servers
  Depending on your environment, you can reduce the default setting of the `PARALLEL_MIN_SERVERS` parameter.
- About Recovery Catalog Upgrade After Upgrading Oracle Database
  If you use a version of the recovery catalog schema that is older than that required by the RMAN client, then you must upgrade it.
- Upgrading Statistics Tables Created by the DBMS_STATS Package After Upgrading Oracle Database
  If you created statistics tables using the `DBMS_STATS.CREATE_STAT_TABLE` procedure, then upgrade these tables by running `DBMS_STATS.UPGRADE_STAT_TABLE`.
- Upgrading Externally Authenticated SSL Users After Upgrading Oracle Database
  If you are upgrading from Oracle9*i* Release 2 (9.2) or Oracle Database 10*g* Release 1 (10.1), and you are using externally authenticated SSL users, then you must run the SSL external users conversion (`extusrupgrade`) script to upgrade those users.
- Configuring the FTP and HTTP Ports and HTTP Authentication for Oracle XML DB
  Oracle Database Configuration Assistant (DBCA) does not configure ports for Oracle XML DB on Oracle Database 12c and later releases. Upgrades use digest authentication.
- Install Oracle Text Supplied Knowledge Bases After Upgrading Oracle Database
  After an Oracle Database upgrade, all user extensions to the Oracle Text supplied knowledge bases must be regenerated.
- Rebuild Oracle Text Indexes Using AUTO_LEXER
  If you are upgrading Oracle Text from releases earlier than Oracle Database 12*c*, then review this topic.
- Update Oracle Application Express Configuration After Upgrading Oracle Database
  Oracle Application Express affects upgrade procedures, depending on the Oracle Application Express release and your database installation type.
- Replace the DEMO Directory in Read-Only Oracle Homes
  After upgrading Read-Only Oracle homes, make a copy of the earlier release Oracle Database `demo` directory, and replace the `demo` directory in the Read-Only Oracle home with the new release `demo` directory.
- Configure Access Control Lists (ACLs) to External Network Services
  Oracle Database 12*c* and later releases include fine-grained access control to the `UTL_TCP`, `UTL_SMTP`, `UTL_MAIL`, `UTL_HTTP`, or `UTL_INADDR` packages.
- Enabling Oracle Database Vault After Upgrading Oracle Database
  Depending on your target database release, you can be required to disable Oracle Database Vault to complete an Oracle Database upgrade.
- Check for the SQLNET.ALLOWED_LOGON_VERSION Parameter Behavior
  Connections to Oracle Database from clients earlier than release 10g fail with the error `ORA-28040: No matching authentication protocol`.

Setting Environment Variables on Linux and Unix Systems After Manual Upgrades

If you performed a manual upgrade of Oracle Database, then you must ensure that the required operating system environment variables point to the directories of the new Oracle Database release.

Confirm that the following environment variables point to the directories of the new Oracle home:

- `ORACLE_HOME`
- `PATH`

> **Note:**
>
> DBUA automatically makes necessary changes to Oracle environment variables.

Related Topics

- *Oracle Database Administrator's Guide*

Recompiling All Invalid Objects

To identify and recompile invalid objects on the CDB and PDBs, use the `catcon` utility to run `utlrp.sql` after you install, patch, or upgrade a database.

Oracle recommends that you use the `catcon.pl` utility to run `utlrp.sql` on all containers in your container database (CDB). The `utlrp.sql` script recompiles all invalid objects. Run the script immediately after installation, to ensure that users do not encounter invalid objects.

1. Change directory to *`Oracle_home`*`/rdbms/admin`. For example

   ```
   $ cd $ORACLE_HOME/rdbms/admin
   ```

2. Use the `catcon.pl` script in the Oracle home to run `utlrp.sql`. For example:

   ```
   $ORACLE_HOME/perl/bin/perl catcon.pl --n 1 --e --b utlrp --d
   '''.''' utlrp.sql
   ```

   Note the following conditions of this use case:

   - `--n` parameter: is set to 1, so the script runs each PDB recompilation in sequence.
   - `--e` parameter: turns echo on.
   - `--b` parameter: Sets the log file base name. It is set to `utlrp`.

Expect a time delay for the serial recompilation of PDBs to complete. Depending on the number of PDBs that you are upgrading, the recompilation can extend significantly beyond the time required for the upgrade scripts to complete.

The `utlrp.sql` script automatically recompiles invalid objects in either serial or parallel recompilation, based on both the number of invalid objects, and on the number of CPUs available. CPUs are calculated using the number of CPUs (`cpu_count`) multiplied by the number of threads for each CPU (`parallel_threads_per_cpu`). On Oracle Real Application Clusters (Oracle RAC), this number is added across all Oracle RAC nodes.

For more information about catcon utility syntax and options, refer to *Oracle Multitenant Administrator's Guide*.

Related Topics

- Syntax and Parameters for catcon.pl

Recompiling All Invalid Objects on Multitenant Architecture Databases

On multitenant architecture Oracle Databases, after upgrading. Oracle recommends that you recompile invalid objects using the `catcon` Perl script.

Use the `catcon.pl` utility to run `utlrp.sql` on all containers in your container database (CDB).

The `catcon.pl` script runs utlrp.sql from the `$ORACLE_HOME/rdbms/admin` directory. The script recompiles any remaining stored PL/SQL and Java code.

Example 6-1 Running the utlrp.sql Script On All Containers in the CDB With the CATCON Utility

```
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlrp -d '''.''' 
utlrp.sql
```

Note the following conditions of this use case:

- The `-n` parameter is set to 1, so the script runs each PDB recompilation in sequence.
- Expect a time delay for the serial recompilation of PDBs to complete. Depending on the number of PDBs that you are upgrading, the recompilation can extend significantly beyond the time required for the upgrade scripts to complete.

Track Invalid Object Recompilation Progress

Use these SQL queries to track the progress of `utlrp.sql` script recompilation of invalid objects.

Oracle recommends that you run the `utlrp.sql` script after upgrade to recompile invalid objects. You can run SQL queries to monitor the script.

Example 6-2 Number of Invalid Objects Remaining

Enter this query to return the number of remaining invalid objects. This number decreases over time as the `utlrp.sql` script runs.

```
SELECT COUNT(*) FROM obj$ WHERE status IN (4, 5, 6);
```

Example 6-3 Number of Objects Recompiled

Enter this query to return the number of objects that `utlrp.sql` has compiled. This number increases over time as the script runs.

```
SELECT COUNT(*) FROM UTL_RECOMP_COMPILED;
```

Example 6-4 Number of Objects Recompiled with Errors

Enter this query to return the number of objects that `utlrp.sql` has compiled with errors.

```
select COUNT(DISTINCT(obj#)) "OBJECTS WITH ERRORS" from
utl_recomp_errors;
```

If the number is higher than expected, then examine the error messages reported with each object. If you see errors due to system misconfiguration or resource constraints, then fix the cause of these errors, and run `utlrp.sql` again.

Setting oratab and Scripts to Point to the New Oracle Location After Upgrading Oracle Database

You must set scripts to point to the new Oracle home location.

After you upgrade Oracle Database to a new release, you must ensure that your `oratab` file and any client scripts that set the value of *`ORACLE_HOME`* point to the new Oracle home that is created for the new Oracle Database release. DBUA automatically points `oratab` to the new Oracle home. However, you must check client scripts regardless of the method you use to upgrade.

If you upgrade your database manually, then you must log in as the Oracle installation owner for the new Oracle Database release, and update the `oratab` file manually. The location of the `oratab` file can vary, depending on your operating system.

See Also:

Oracle Database Administrator's Guide for information about setting operating system environment variables

My Oracle Support: Find or Create Oratab File (Doc ID 394251.1)

https://support.oracle.com/rs?type=doc&id=394251.1

Check PL/SQL Packages and Dependent Procedures

It is possible that packages that you installed in the earlier release Oracle Database are not available in the new release, which can affect applications.

After the upgrade, check to ensure that any packages that you have used in your own scripts, or that you call from your scripts, are available in the new release. Testing procedures dependent on packages should be part of your upgrade plan.

Code in database applications can reference objects in the connected database. For example, Oracle Call Interface (OCI) and precompiler applications can submit anonymous PL/SQL blocks. Triggers in Oracle Forms applications can reference a schema object. Such applications are dependent on the schema objects they reference. Dependency management techniques vary, depending on the development environment. Oracle Database does not automatically track application dependencies.

Related Topics

- *Oracle Database Administrator's Guide*

Upgrading Tables Dependent on Oracle-Maintained Types

Starting with Oracle Database 12*c* Release 2 (12.2) and later releases, you must manually upgrade user tables that depend on Oracle-Maintained types.

If your database has user tables that are dependent on Oracle-Maintained types (for example, `AQ` queue tables), then run the `utluptabdata.sql` command after the upgrade to carry out `ALTER TABLE UPGRADE` on any user tables affected by changes in Oracle-Maintained types. This change in behavior enables user tables to remain in `READ ONLY` state during an upgrade. Users are prevented from logging into applications using SYSDBA privileges (`AS SYSDBA`), and changing application tables that are dependent on Oracle-Maintained types.

To identify tables that you need to upgrade after the database upgrade completes, connect to the database `AS SYSDBA`, and run the following query:

```
COLUMN owner FORMAT A30
COLUMN table_name FORMAT A30
SELECT DISTINCT owner, table_name
FROM dba_tab_cols
WHERE data_upgraded = 'NO'
ORDER BY 1,2;
```

This query lists all tables that are not listed as `UPGRADED`. However, the `utluptabdata.sql` script only upgrades tables that depend on Oracle-Maintained types. If any tables are listed by the query, then run the `utluptabdata.sql` script to perform `ALTER TABLE UPGRADE` commands on dependent user tables, so that these Oracle-Maintained types are upgraded to the latest version of the type.

You must run the `utluptabdata.sql` script either with a user account with `ALTER` privileges for all of the tables dependent on Oracle-Maintained types, or with a user granted the SYSDBA system privileges, and that is logged in `AS SYSDBA`.

When the parameter SERVEROUTPUT is set to ON, the utluptabdata.sql script displays the names of all upgraded tables, and lists any error encountered during the table upgrade. To set the server output to ON, run the following command:

```
SET SERVEROUTPUT ON
@utluptabdata.sql
```

Adjusting Minimum and Maximum for Parallel Execution Servers

Depending on your environment, you can reduce the default setting of the PARALLEL_MIN_SERVERS parameter.

In Oracle Database 12*c*, the default for PARALLEL_MIN_SERVERS changed from 0 to a value that was provided based on your hardware platform. This change was made to provide sufficient minimal support for parallel execution. If you find that the new default setting is too high, then adjust the setting for your requirements. The default for PARALLEL_MAX_SERVERS has not changed. If the default in your old environment is unchanged, then you do not need to take further action.

See Also:

Oracle Database Reference for information about PARALLEL_MIN_SERVERS

About Recovery Catalog Upgrade After Upgrading Oracle Database

If you use a version of the recovery catalog schema that is older than that required by the RMAN client, then you must upgrade it.

See Also:

- *Oracle Database Backup and Recovery User's Guide* for information on managing an RMAN recovery catalog
- *Oracle Database Backup and Recovery User's Guide* for complete information about upgrading the recovery catalog and the UPGRADE CATALOG command

Upgrading Statistics Tables Created by the DBMS_STATS Package After Upgrading Oracle Database

If you created statistics tables using the DBMS_STATS.CREATE_STAT_TABLE procedure, then upgrade these tables by running DBMS_STATS.UPGRADE_STAT_TABLE.

In the following example, `green` is the owner of the statistics table and `STAT_TABLE` is the name of the statistics table.

```
EXECUTE DBMS_STATS.UPGRADE_STAT_TABLE('green', 'stat_table');
```

Perform this procedure for each statistics table.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about the DBMS_STATS package

Upgrading Externally Authenticated SSL Users After Upgrading Oracle Database

If you are upgrading from Oracle9*i* Release 2 (9.2) or Oracle Database 10*g* Release 1 (10.1), and you are using externally authenticated SSL users, then you must run the SSL external users conversion (`extusrupgrade`) script to upgrade those users.

The `extusrupgrade` script has the following syntax, where *`ORACLE_HOME`* is the Oracle database home, *`hostname`* is the name of the host on which the database is running, *`port_no`* is the listener port number, *`sid`* is the system identifier for the database instance, and *`db_admin`* is the database administrative user with privileges to modify user accounts.

```
ORACLE_HOME/rdbms/bin/extusrupgrade --dbconnectstring
hostname:port_no:sid --dbuser db_admin --dbuserpassword
password -a
```

For example:

```
extusrupgrade --dbconnectstring dlsun88:1521:10gR2 --dbuser system --
dbuserpassword manager -a
```

> **Note:**
>
> If you are upgrading from Oracle Database 10*g* Release 2 (10.2) or later, then you are not required to run the `extusrupgrade` script.

> **See Also:**
>
> *Oracle Database Enterprise User Security Administrator's Guide* for more information about the `extusrupgrade` script

Configuring the FTP and HTTP Ports and HTTP Authentication for Oracle XML DB

Oracle Database Configuration Assistant (DBCA) does not configure ports for Oracle XML DB on Oracle Database 12c and later releases. Upgrades use digest authentication.

Oracle recommends that when you configure ports, you also configure the authentication for HTTP for accessing Oracle XML DB Repository to take advantage of improved security features.

Starting with Oracle Database 12c, Oracle enhanced database security by supporting digest authentication. Digest authentication is an industry-standard protocol that is commonly used with the HTTP protocol. It is supported by most HTTP clients. Digest authentication ensures that passwords are always transmitted in a secure manner, even when an encrypted (HTTPS) connection is not in use. Support for digest authentication enables organizations to deploy applications that use Oracle XML DB HTTP, without having to worry about passwords being compromised. Digest authentication support in Oracle XML DB also ensures that the Oracle XML DB HTTP server remains compatible with Microsoft Web Folders WebDAV clients.

After installing or upgrading for the new release, you must manually configure the FTP and HTTP ports for Oracle XML DB as follows:

1. Use `DBMS_XDB_CONFIG.setHTTPPort(`*`HTTP_port_number`*`)` to set the HTTP port for Oracle XML DB:

   ```
   SQL> exec DBMS_XDB_CONFIG.setHTTPPort(port_number);
   ```

2. Use `DBMS_XDB_CONFIG.setFTPPort(`*`FTP_port_number`*`)` to set the FTP port for Oracle XML DB:

   ```
   SQL> exec DBMS_XDB_CONFIG.setFTPPort(FTP_port_number);
   ```

   > **Note:**
   >
   > You can query the port numbers to use for FTP and HTTP in the procedure by using `DBMS_XDB_CONFIG.getFTPPort` and `DBMS_XDB_CONFIG.getHTTPPort` respectively.

3. To see all the used port numbers, query `DBMS_XDB_CONFIG.usedport`.

Install Oracle Text Supplied Knowledge Bases After Upgrading Oracle Database

After an Oracle Database upgrade, all user extensions to the Oracle Text supplied knowledge bases must be regenerated.

Regenerating the user extensions affect all databases installed in the given Oracle home.

After an upgrade, the Oracle Text-supplied knowledge bases that are part of the companion products for the new Oracle Database are not immediately available. Any Oracle Text features dependent on the supplied knowledge bases that were available before the upgrade do not function after the upgrade. To re-enable such features, you must install the Oracle Text supplied knowledge bases from the installation media for the new Oracle Database release.

See Also:

- *Oracle Text Application Developer's Guide* for information about Oracle Text-supplied knowledge bases
- *Oracle Database Installation Guide* for companion products

Rebuild Oracle Text Indexes Using AUTO_LEXER

If you are upgrading Oracle Text from releases earlier than Oracle Database 12*c*, then review this topic.

After you complete your upgrade to the new Oracle Database release, if you use Oracle Text indexes created with AUTO_LEXER, then you must rebuild the indexes for your queries to work.

In addition, you must rebuild indexes that have the following INDEX_STEMS types of BASIC_LEXER set:

- ARABIC
- BOKMAL
- CATALAN
- CROATIAN
- CZECH
- DANISH
- ERIVATIONAL_NEW
- DUTCH_NEW
- ENGLISH_NEW
- FINNISH
- FRENCH_NEW

- GERMAN_NEW
- GREEK
- HEBREW
- HUNGARIAN
- ITALIAN_NEW
- NYNORSK
- POLISH
- PORTUGUESE
- ROMANIAN
- RUSSIAN
- SERBIAN
- SLOVAK
- SLOVENIAN
- SPANISH_NEW
- SWEDISH

Update Oracle Application Express Configuration After Upgrading Oracle Database

Oracle Application Express affects upgrade procedures, depending on the Oracle Application Express release and your database installation type.

If the Oracle Database release that you upgrade includes Oracle Application Express release 3.2 or later, then you do not need to carry out additional configuration after upgrading to the new Oracle Database release. However, if Oracle Application Express is in the registry, so that Oracle Application Express is included in the upgrade, then set the `open_cursors` parameter to a minimum of 200.

If the Oracle Database you upgrade is an Oracle Express Edition database, then it contains an earlier release of Oracle Application Express that is tailored for the Oracle Express Edition environment. The latest Oracle Application Express release is automatically installed during the upgrade. You must complete a series of postinstallation steps to configure Oracle Application Express for use with the new Oracle Database release.

See Also:

- *Oracle Application Express Installation Guide* for postinstallation tasks for Oracle Application Express
- `http://www.oracle.com/technetwork/developer-tools/apex/overview/index.html`

Replace the DEMO Directory in Read-Only Oracle Homes

After upgrading Read-Only Oracle homes, make a copy of the earlier release Oracle Database `demo` directory, and replace the `demo` directory in the Read-Only Oracle home with the new release `demo` directory.

Oracle Database 18c and later releases contain a product demonstration directory in the file path *`Oracle_home`*`/rdbms/demo`. These directories include examples and product demonstrations that are specific to the options and features for each Oracle Database release, some of which you can add to after upgrade by installing Oracle Database Examples. In your earlier release, if you downloaded and worked with the earlier release demonstration files, then you have two problems: you want to save your earlier release work for review and testing with the new release, and you want to obtain refreshes of the demonstrations that are specific to the new release.

After upgrading the Oracle home, and downloading and doing any other work you want to do with the new demonstration files, you can then refresh your old demonstration files.

Example 6-5 Copying the Earlier Release Demo Directory and Refreshing the Demonstrations in the Read-Only Oracle Home

After the upgrade, use this procedure to save any work in your earlier `demo` directory in the Read-Only Oracle home, and and replace the earlier release `demo` directory with the new release `demo` directory:

1. Log in as the Oracle software owner user (`oracle`).
2. Check if the `rdbms/demo` directory is copied to the Read Only Oracle home.

   In this example, the environment variable `ORACLE_BASE_HOME` is defined as the path to the Read-Only Oracle home.

   Linux and Unix platforms:

   ```
   $ ls -l -d $ORACLE_BASE_HOME/rdbms/demo
   /u01/app/oracle/product/19.0.0/dbhome_1/rdbms/demo
   ```

   Microsoft Windows platforms

   ```
   ls -l -d %ORACLE_BASE_HOME%\rdbms\demo
   %ORACLE_BASE_HOME%\rdbms\demo
   ```

3. Change directory to the Read-Only Oracle home, and make a copy, where `demo.old_release18` is the name you give to your earlier release demonstration files:

   ```
   cd $ORACLE_BASE_HOME/rdbms
   mv demo demo.old_release18
   ```

4. Copy the new `demo` directory from the upgraded Oracle home to the Read-Only Oracle home.

   In this example, the environment variable ORACLE_HOME is defined as the new release Oracle home.

Linux and Unix:

```
cp -r $ORACLE_HOME/rdbms/demo demo
```

Microsoft Windows

```
xcopy c:\%ORACLE_HOME%\rdbms\demo c:%ORACLE_BASE_HOME%\rdbms\demo /E
```

Configure Access Control Lists (ACLs) to External Network Services

Oracle Database 12*c* and later releases include fine-grained access control to the `UTL_TCP`, `UTL_SMTP`, `UTL_MAIL`, `UTL_HTTP`, or `UTL_INADDR` packages.

If you have applications that use these packages, then after upgrading Oracle Database you must configure network access control lists (ACLs) in the database before the affected packages can work as they did in earlier releases. Without the ACLs, your applications can fail with the error "ORA-24247: network access denied by access control list (ACL)."

> **See Also:**
>
> *Oracle Database Security Guide* for more complicated situations, such as connecting some users to host A and other users to host B

Enabling Oracle Database Vault After Upgrading Oracle Database

Depending on your target database release, you can be required to disable Oracle Database Vault to complete an Oracle Database upgrade.

- Upgrading Oracle Database Without Disabling Oracle Database Vault
  If your target Oracle Database release is 12.2 or later, then you can upgrade without disabling Oracle Database Vault.
- Common Upgrade Scenarios with Oracle Database Vault
  The requirements to enable Oracle Database Vault after upgrades change, depending on your source Oracle Database release.

Upgrading Oracle Database Without Disabling Oracle Database Vault

If your target Oracle Database release is 12.2 or later, then you can upgrade without disabling Oracle Database Vault.

If you have Oracle Database Vault enabled in your source Oracle Database release, then you can upgrade Oracle Database to Oracle Database 18c and later releases without first disabling Oracle Database Vault. After the upgrade, if your source Oracle Database release is Oracle Database 12c release 1 (12.1) or later, then Oracle Database Vault is enabled with the same enforcement settings that you had in place before the upgrade. For example, if your source database is Oracle Database release 12.1, and Oracle Database Vault was disabled in that release, then it remains disabled

after you upgrade. If your source Oracle Database release 12.1 database had Oracle Database Vault enabled before the upgrade, then Oracle Database Vault is enabled after the upgrade.

If you manually disable Oracle Database Vault before the upgrade, then you must enable Oracle Database Vault manually after the upgrade.

If you did not have Oracle Database Vault enabled before the upgrade, then you can enable it manually after the upgrade.

Enable Oracle Database Vault in the upgraded database by using the procedure `dvsys.dbms_macadm.enable_dv()`. Run this procedure with a user account that is granted `DV_OWNER`. After you run the procedure, restart the database instance so that the procedure takes effect.

Related Topics

- *Oracle Database Vault Administrator's Guide*

Common Upgrade Scenarios with Oracle Database Vault

The requirements to enable Oracle Database Vault after upgrades change, depending on your source Oracle Database release.

- Upgrades from Oracle Database 11*g* release 2 (11.2) or earlier: After the upgrade, Oracle Database Vault is disabled by default.
- Upgrades from Oracle Database 12*c* release 1 (12.1) or later: After the upgrade, Oracle Database Vault has the same enforcement status that you had in place before the upgrade.

Table 6-1 Common Oracle Database Vault Upgrade Scenarios and Upgrade Preparation Tasks

Source Database Release	Target Database Release	Do you need to disable Database Vault Before Upgrade	What is Database Vault Status After Upgrade
11.2 or earlier	12.1	Yes	Disabled. You need to enable Database Vault manually after the upgrade.
11.2.or earlier	12.2, 18.1 and later	No	Disabled. You need to enable Database Vault manually after the upgrade.
12.1, 12.2, 18.1, and later	12.2, 18.1 and later	No	Database Vault has the same enforcement status that you had in place before the upgrade.

Check for the SQLNET.ALLOWED_LOGON_VERSION Parameter Behavior

Connections to Oracle Database from clients earlier than release 10g fail with the error `ORA-28040: No matching authentication protocol`.

Starting with Oracle Database 18c, the default value for the `SQLNET.ALLOWED_LOGON_VERSION` parameter changed from 11 in Oracle Database 12c (12.2) to 12 in Oracle Database 18c and later releases. The use of this parameter is deprecated.

`SQLNET.ALLOWED_LOGON_VERSION` is now replaced with the `SQLNET.ALLOWED_LOGON_VERSION_SERVER` and `SQLNET.ALLOWED_LOGON_VERSION_CLIENT` parameters. If you have not explicitly set the `SQLNET.ALLOWED_LOGON_VERSION_SERVER` parameter in the upgraded database, then connections from clients earlier than release 10g fail with the error `ORA-28040: No matching authentication protocol`. For better security, check the password verifiers of your database users, and then configure the database to use the correct password verifier by setting the `SQLNET.ALLOWED_LOGON_VERSION_SERVER` and `SQLNET.ALLOWED_LOGON_VERSION_CLIENT` parameters.

If you have password-protected roles (secure roles) in your existing database, and if you upgrade to Oracle Database 18c and later releases with the default `SQLNET.ALLOWED_LOGON_VERSION_SERVER` setting of 12, because those secure roles only have release 10g verifiers, then the password for each secure role must be reset by the administrator so that the secure roles can remain usable after the upgrade.

See Also:

- *Oracle Database Security Guide* for information about ensuring against password security threats
- *Oracle Database Security GuideOracle Database Security Guide* for information about setting the password versions of users

Recommended and Best Practices to Complete After Upgrading Oracle Database

Oracle recommends that you complete these good practices guidelines for updating Oracle Database. Except where noted, these practices are recommended for all type of upgrades (Manual, DBUA or AutoUpgrade).

- Back Up the Database
  Oracle strongly recommends that you at least perform a level 1 backup, or if time allows, perform a level 0 backup.
- Running Postupgrade Fixup Scripts
  Review this procedure to understand how to use the `postupgrade_fixups.sql` scripts for CDB and Non-CDB databases.

- Gathering Dictionary Statistics After Upgrading
  To help to assure good performance, use this procedure to gather dictionary statistics after completing your upgrade.
- Regathering Fixed Objects Statistics with DBMS_STATS
  If you do not use AutoUpgrade, then after an upgrade, or after other database configuration changes, Oracle strongly recommends that you regather fixed object statistics after you have run representative workloads on Oracle Database.
- Reset Passwords to Enforce Case-Sensitivity
  For upgraded databases, improve security by using case-sensitive passwords for default user accounts and user accounts.
- Finding and Resetting User Passwords That Use the 10G Password Version
  For better security, find and reset passwords for user accounts that use the `10G` password version so that they use later, more secure password versions.
- Understand Oracle Grid Infrastructure, Oracle ASM, and Oracle Clusterware
  Oracle Clusterware and Oracle Automatic Storage Management (Oracle ASM) are both part of an Oracle Grid Infrastructure installation.
- Oracle Grid Infrastructure Installation and Upgrade and Oracle ASM
  Oracle ASM is installed with Oracle Grid Infrastructure.
- Add New Features as Appropriate
  Review new features as part of your database upgrade plan.
- Develop New Administrative Procedures as Needed
  Plan a review of your scripts and procedures, and change as needed.
- Migrating From Rollback Segments To Automatic Undo Mode
  If your database release is earlier than Oracle Database 11*g*, then you must migrate the database that is being upgraded from using rollback segments (manual undo management) to automatic undo management.
- Migrating Tables from the LONG Data Type to the LOB Data Type
  You can use the `ALTER TABLE` statement to change the data type of a `LONG` column to `CLOB` and that of a `LONG RAW` column to `BLOB`.
- Migrate Your Upgraded Oracle Databases to Use Unified Auditing
  To use the full facilities of unified auditing, you must manually migrate to unified auditing.
- Identify Oracle Text Indexes for Rebuilds
  You can run a script that helps you to identify Oracle Text index indexes with token tables that can benefit by being rebuilt after upgrading to the new Oracle Database release..
- Dropping and Recreating DBMS_SCHEDULER Jobs
  If DBMS_SCHEDULER jobs do not function after upgrading from an earlier release, drop and recreate the jobs.
- Transfer Unified Audit Records After the Upgrade
  Review these topics to understand how you can obtain better performance after you upgrade and migrate to unified auditing
- About Testing the Upgraded Production Oracle Database
  Repeat tests on your production database that you carried out on your test database to ensure applications operate as expected.

- Upgrading the Time Zone File Version After Upgrading Oracle Database
  If the Pre-Upgrade Information Tool instructs you to upgrade the time zone files after completing the database upgrade, then use the `DBMS_DST` PL/SQL package to upgrade the time zone file.

Back Up the Database

Oracle strongly recommends that you at least perform a level 1 backup, or if time allows, perform a level 0 backup.

Related Topics

- Backing Up the Database

Running Postupgrade Fixup Scripts

Review this procedure to understand how to use the `postupgrade_fixups.sql` scripts for CDB and Non-CDB databases.

The postupgrade fixup scripts are generated when you run the Pre-Upgrade Information Tool (`preupgrade.jar`). Run the postupgrade scripts any time after completing an upgrade. For both Container Databases (CDBs) with pluggable databases (PDBs), and for Non-CDB databases, the postupgrade fixup scripts provide general warnings, errors, and informational recommendations.

You can run the script either by using the `catcon.pl` utility, or by using SQL*Plus.

The location of the postupgrade SQL scripts and log files depends on how you set output folders, or define the Oracle base environment variable. The postupgrade fixup scripts are placed in the same directory path as the preupgrade fixup scripts.

If you specify an output directory by using the `dir` option with the Pre-Upgrade Information Tool, then the output logs and files are placed under that directory in the file path /`cfgtoollogs`/*`dbunique_name`*/`preupgrade`, where *`dbunique_name`* is the name of your source Oracle Database. If you do not specify an output directory when you run the Pre-Upgrade Information Tool, then the output is directed to one of the following default locations:

- If you do not specify an output directory with `DIR`, but you have set an Oracle base environment variable, then the generated scripts and log files are created in the following file path:

  `Oracle-base``/cfgtoollogs/`*`dbunique_name`*`/preupgrade`

- If you do not specify an output directory, and you have not defined an Oracle base environment variable, then the generated scripts and log files are created in the following file path:

  `Oracle-home``/cfgtoollogs/`*`dbunique_name`*`/preupgrade`

The postupgrade fixup scripts that the Pre-Upgrade Information Tool creates depend on whether your source database is a Non-CDB database, or a CDB database:

- Non-CDB: `postupgrade_fixups.sql`
- CDB: Two different sets of scripts:
  1. `postupgrade_fixups.sql`: A consolidated script for all PDBs

2. Multiple `postupgrade_fixups_`*pdbname*`.sql` scripts, where*pdbname* is the name of the PDB for which a script is generated: Individual scripts, which you run on specific PDBs.

Example 6-6 Example of Spooling Postupgrade Fixup Results for a Non-CDB Oracle Database

Set the system to spool results to a log file so you can read the output. However, do not spool to the `admin` directory:

```
SQL> SPOOL postupgrade.log
SQL> @postupgrade_fixups.sql
SQL> SPOOL OFF
```

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

```
SQL> SPOOL OFF
```

Example 6-7 Examples of Running Postupgrade Fixups Using catcon.pl On Linux, UNIX and Windows Systems

In the examples in this section, the `catcon` command runs `postupgrade_fixups.sql` in all the containers of a CDB database. Before you run this command, you must ensure that the operating system environment variables ORACLE_HOME and ORACLE_SID are set for UNIX and Linux environments, and that their equivalents are set for Windows environments. In the following examples, `sales1` is the unique name for the target database.

Linux and UNIX environments:

```
$ORACLE_HOME/perl/bin/perl -I$ORACLE_HOME/perl/lib -I$ORACLE_HOME/rdbms/
admin $ORACLE_HOME/rdbms/admin/catcon.pl -l /home/oracle/here/ -b
postupg /cfgtoollogs/sales1/preupgrade/postupgrade_fixups.sql
```

Windows environments:

```
%ORACLE_HOME%\perl\bin\perl –I%ORACLE_HOME%\perl\lib –I%ORACLE_HOME%
\rdbms\admin %ORACLE_HOME%\rdbms\admin\catcon.pl -l c:\tmp\logdir\ -b
postupg c:\cfgtoollogs\sales1\preupgrade\postupgrade_fixups.sql
```

In this Windows example, the command option `-l` creates logs under the file path `c:/tmp/logdir/`, and places the output from the postupgrade scripts into that directory. The option `-b` sets the prefix `postupg` on the output files.

> **Note:**
>
> You must enter the `catcon` parameters in the order that they are shown in these examples.

Gathering Dictionary Statistics After Upgrading

To help to assure good performance, use this procedure to gather dictionary statistics after completing your upgrade.

Oracle recommends that you gather dictionary statistics both before and after upgrading the database, because Data Dictionary tables are modified and created during the upgrade. With Oracle Database 12*c* release 2 (12.2) and later releases, you gather statistics as a manual procedure after the upgrade, when you bring the database up in normal mode.

- Non-CDB Oracle Database: Oracle recommends that you use the `DBMS_STATS.GATHER_DICTIONARY_STATS` procedure to gather these statistics. For example, enter the following SQL statement:

  ```
  SQL> EXEC DBMS_STATS.GATHER_DICTIONARY_STATS;
  ```

- CDB: Oracle recommends that you use `catcon` to gather Data Dictionary statistics across the entire multitenant architecture

  To gather dictionary statistics for all PDBs in a container database, use the following syntax

  ```
  $ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -l /tmp
  -b gatherstats -- --x"exec dbms_stats.gather_dictionary_stats"
  ```

  To gather dictionary statistics on a particular PDB, use syntax similar to the following:

  ```
  $ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -
  l /tmp -c
  'SALES1' -b gatherstats -- --x"exec
  dbms_stats.gather_dictionary_stats"
  ```

  In the preceding example the `-c SALES1` option specifies a PDB inclusion list for the command that you run, specifying the database named `SALES1`. The option `-b gatherstats` specifies the base name for the logs. The option `--x` specifies the SQL command that you want to execute. The SQL command itself is inside the quotation marks.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

Regathering Fixed Objects Statistics with DBMS_STATS

If you do not use AutoUpgrade, then after an upgrade, or after other database configuration changes, Oracle strongly recommends that you regather fixed object statistics after you have run representative workloads on Oracle Database.

> **Note:**
>
> If you upgraded using the AutoUpgrade utility, then AutoUpgrade automatically updates fixed object statistics during the upgrade. You do not need to perform this task.

Fixed objects are the `X$` tables and their indexes. `V$` performance views are defined through `X$` tables. Gathering fixed object statistics is valuable for database performance, because these statistics help the optimizer to generate good execution plans, which can improve database performance. Failing to obtain representative statistics can lead to suboptimal execution plans, which can cause significant performance problems.

Ensure that your database has run representative workloads, and then gather fixed objects statistics by using the `DBMS_STATS.GATHER_FIXED_OBJECTS_STATS` PL/SQL procedure. `DBMS_STATS.GATHER_FIXED_OBJECTS_STATS` also displays recommendations for removing all hidden or underscore parameters and events from the `INIT.ORA` or `SPFILE`.

Because of the transient nature of `X$` tables, you must gather fixed objects statistics when there is a representative workload on the system. If you cannot gather fixed objects statistics during peak load, then Oracle recommends that you do it after the system is in a runtime state, and the most important types of fixed object tables are populated.

To gather statistics for fixed objects, run the following PL/SQL procedure:

```
SQL> execute dbms_stats.gather_fixed_objects_stats;
```

Reset Passwords to Enforce Case-Sensitivity

For upgraded databases, improve security by using case-sensitive passwords for default user accounts and user accounts.

For greater security, Oracle recommends that you enable case sensitivity in passwords. Case sensitivity increases the security of passwords by requiring that users enter both the correct password string, and the correct case for each character in that string. For example, the password `hPP5620qr` fails if it is entered as `hpp5620QR` or `hPp5620Qr`.

To secure your database, create passwords in a secure fashion. If you have default passwords in your database, then change these passwords. By default, case sensitivity is enforce when you change passwords. Every password should satisfy the Oracle recommended password requirements, including passwords for predefined user accounts.

For new databases created after the upgrade, there are no additional tasks or management requirements.

Existing Database Requirements and Guidelines for Password Changes

- If the default security settings for Oracle Database 12*c* release 1 (12.1) and later are in place, then passwords must be at least eight characters, and passwords such as `welcome` and `oracle` are not allowed.
- The `IGNORECASE` parameter is deprecated. Do not use this parameter.
- For existing databases, to take advantage of password case-sensitivity, you must reset the passwords of existing users during the database upgrade procedure. Reset the password for each existing database user with an `ALTER USER` statement.
- Query the `PASSWORD_VERSIONS` column of `DBA_USERS` to find the `USERNAME` of accounts that only have the 10G password version, and do not have either the `11G` or the `12C` password version. Reset the password for any account that has only the `10G` password version.

See Also:

- *Oracle Database Security Guide* for more information about password case sensitivity
- *Oracle Database Security Guide* for more information about password strength

Finding and Resetting User Passwords That Use the 10G Password Version

For better security, find and reset passwords for user accounts that use the `10G` password version so that they use later, more secure password versions.

Finding All Password Versions of Current Users

You can query the `DBA_USERS` data dictionary view to find a list of all the password versions configured for user accounts.

For example:

```
SELECT USERNAME,PASSWORD_VERSIONS FROM DBA_USERS;

USERNAME                       PASSWORD_VERSIONS
------------------------------ -----------------
JONES                          10G 11G 12C
ADAMS                          10G 11G
CLARK                          10G 11G
PRESTON                        11G
BLAKE                          10G
```

The PASSWORD_VERSIONS column shows the list of password versions that exist for the account. 10G refers to the earlier case-insensitive Oracle password version, 11G refers to the SHA-1-based password version, and 12C refers to the SHA-2-based SHA-512 password version.

- User jones: The password for this user was reset in Oracle Database 12*c* Release 12.1 when the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter setting was 8. This enabled all three password versions to be created.
- Users adams and clark: The passwords for these accounts were originally created in Oracle Database 10*g* and then reset in Oracle Database 11*g*. The Oracle Database 11*g* software was using the default SQLNET.ALLOWED_LOGON_VERSION setting of 8 at that time. Because case insensitivity is enabled by default, their passwords are now case sensitive, as is the password for preston.
- User preston: This account was imported from an Oracle Database 11*g* database that was running in Exclusive Mode (SQLNET.ALLOWED_LOGON_VERSION = 12).
- User blake: This account still uses the Oracle Database 10*g* password version. At this stage, user blake is prevented from logging in.

Resetting User Passwords That Use the 10G Password Version

For better security, remove the 10G password version from the accounts of all users. In the following procedure, to reset the passwords of users who have the 10G password version, you must temporarily relax the SQLNET.ALLOWED_LOGON_VERSION_SERVER setting, which controls the ability level required of clients before login can be allowed. Relaxing the setting enables these users to log in and change their passwords, and hence generate the newer password versions in addition to the 10G password version. Afterward, you can set the database to use Exclusive Mode and ensure that the clients have the O5L_NP capability. Then the users can reset their passwords again, so that their password versions no longer include 10G, but only have the more secure 11G and 12C password versions.

1. Query the DBA_USERS view to find users who only use the 10G password version.

```
SELECT USERNAME FROM DBA_USERS
WHERE ( PASSWORD_VERSIONS = '10G '
OR PASSWORD_VERSIONS = '10G HTTP ')
AND USERNAME <> 'ANONYMOUS';
```

2. Configure the database so that it does not run in Exclusive Mode, as follows:
   a. Edit the SQLNET.ALLOWED_LOGON_VERSION_SERVER setting in the sqlnet.ora file so that it is more permissive than the default. For example:

      ```
      SQLNET.ALLOWED_LOGON_VERSION_SERVER=11
      ```

   b. Restart the database.
3. Expire the users that you found when you queried the DBA_USERS view to find users who only use the 10G password version.

   You must expire the users who have only the 10G password version, and do not have one or both of the 11G or 12C password versions.

   For example:

   ```
   ALTER USER username PASSWORD EXPIRE;
   ```

4. Ask the users whose passwords you expired to log in.

When the users log in, they are prompted to change their passwords. The database generates the missing 11G and 12C password versions for their account, in addition to the 10G password version. The 10G password version continues to be present, because the database is running in the permissive mode.

5. Ensure that the client software with which the users are connecting has the O5L_NP ability.

   All Oracle Database release 11.2.0.3 and later clients have the O5L_NP ability. If you have an earlier Oracle Database client, then you must install the CPUOct2012 patch.

6. After all clients have the O5L_NP capability, set the security for the server back to Exclusive Mode, as follows:

   a. Remove the SEC_CASE_SENSITIVE_LOGON parameter setting from the instance initialization file, or set SEC_CASE_SENSITIVE_LOGON to TRUE.

      ```
      SEC_CASE_SENSITIVE_LOGON = TRUE
      ```

   b. Remove the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter from the server sqlnet.ora file, or set the value of SQLNET.ALLOWED_LOGON_VERSION_SERVER in the server sqlnet.ora file back to 12, to set it to an Exclusive Mode.

      ```
      SQLNET.ALLOWED_LOGON_VERSION_SERVER = 12
      ```

   c. Restart the database.

7. Find the accounts that still have the 10G password version.

   ```
   SELECT USERNAME FROM DBA_USERS
   WHERE PASSWORD_VERSIONS LIKE '%10G%'
   AND USERNAME <> 'ANONYMOUS';
   ```

8. Expire the accounts that still have the 10G password version.

   ```
   ALTER USER username PASSWORD EXPIRE;
   ```

9. Ask these users to log in to their accounts.

   When the users log in, they are prompted to reset their passwords. The database then generates only the 11G and 12C password versions for their accounts. Because the database is running in Exclusive Mode, the 10G password version is no longer generated.

10. Rerun the following query:

    ```
    SELECT USERNAME FROM DBA_USERS
    WHERE PASSWORD_VERSIONS LIKE '%10G%'
    AND USERNAME <> 'ANONYMOUS';
    ```

    If this query does not return any results, then it means that no user accounts have the 10G password version. Hence, the database is running in a more secure mode than in previous releases.

Understand Oracle Grid Infrastructure, Oracle ASM, and Oracle Clusterware

Oracle Clusterware and Oracle Automatic Storage Management (Oracle ASM) are both part of an Oracle Grid Infrastructure installation.

If Oracle Grid Infrastructure is installed for a single server, then it is deployed as an Oracle Restart installation with Oracle ASM. If Oracle Grid Infrastructure is installed for a cluster, then it is deployed as an Oracle Clusterware installation with Oracle ASM.

Oracle Restart enhances the availability of Oracle Database in a single-instance environment. If you install Oracle Restart, and there is a temporary failure of any part of the Oracle Database software stack, including the database, listener, and Oracle ASM instance, Oracle Restart automatically restarts the failed component. In addition, Oracle Restart starts all these components when the database host computer is restarted. The components are started in the proper order, taking into consideration the dependencies among components.

Oracle Clusterware is portable cluster software that enables clustering of single servers so that they cooperate as a single system. Oracle Clusterware also provides the required infrastructure for Oracle RAC. In addition, Oracle Clusterware enables the protection of any Oracle application or any other application within a cluster. In any case Oracle Clusterware is the intelligence in those systems that ensures required cooperation between the cluster nodes.

Oracle Grid Infrastructure Installation and Upgrade and Oracle ASM

Oracle ASM is installed with Oracle Grid Infrastructure.

In earlier releases, Oracle ASM was installed as part of the Oracle Database installation. Starting with Oracle Database release 11.2, Oracle ASM is installed when you install the Grid Infrastructure components. Oracle ASM shares an Oracle home with Oracle Clusterware.

> **See Also:**
>
> *Oracle Grid Infrastructure Installation Guide* for your platform for information about Oracle homes, role-allocated system privileges groups, different installation software owner users, and other changes.

Add New Features as Appropriate

Review new features as part of your database upgrade plan.

Oracle Database New Features Guide describes many of the new features available in the new Oracle Database release. Determine which of these new features can benefit the database and applications. You can then develop a plan for using these features.

It is not necessary to make any immediate changes to begin using your new Oracle Database software. You can choose to introduce new feature enhancements into your database and applications gradually.

> **See Also:**
>
> *Oracle Database New Features Guide*

Develop New Administrative Procedures as Needed

Plan a review of your scripts and procedures, and change as needed.

After familiarizing yourself with the features of the new Oracle Database release, 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.

Migrating From Rollback Segments To Automatic Undo Mode

If your database release is earlier than Oracle Database 11*g*, then you must migrate the database that is being upgraded from using rollback segments (manual undo management) to automatic undo management.

Automatic undo management is the default undo space management mode. The `UNDO_MANAGEMENT` initialization parameter specifies which undo space management mode the system should use:

- If `UNDO_MANAGEMENT` is set to `AUTO` (or if `UNDO_MANAGEMENT` is not set), then the database instance starts in automatic undo management mode.

  A null `UNDO_MANAGEMENT` initialization parameter defaults to automatic undo management mode in Oracle Database 11*g* Release 1 (11.1) and later. In earlier releases it defaults to manual undo management mode. Use caution when upgrading earlier releases.
- If `UNDO_MANAGEMENT` is set to `MANUAL`, then undo space is allocated externally as rollback segments.

1. Set the `UNDO_MANAGEMENT` parameter to `UNDO_MANAGEMENT=MANUAL`.
2. Start the instance again and run through a standard business cycle to obtain a representative workload. Assess the workload, and compute the size of the undo tablespace that you require for automatic undo management.
3. After the standard business cycle completes, run the following function to collect the undo tablespace size, and to help with the sizing of the undo tablespace. You require SYSDBA privileges to run this function.

   ```
   DECLARE
      utbsiz_in_MB NUMBER;
   BEGIN
      utbsiz_in_MB := DBMS_UNDO_ADV.RBU_MIGRATION;
   end;
   /
   ```

   This function runs a PL/SQL procedure that provides information on how to size your new undo tablespace based on the configuration and usage of the rollback segments in your system. The function returns the sizing information directly.
4. Create an undo tablespace of the required size and turn on the automatic undo management by setting `UNDO_MANAGEMENT=AUTO` or by removing the parameter.

5. For Oracle RAC configurations, repeat these steps on all instances.

Migrating Tables from the LONG Data Type to the LOB Data Type

You can use the `ALTER TABLE` statement to change the data type of a `LONG` column to `CLOB` and that of a `LONG RAW` column to `BLOB`.

The `LOB` data types (`BFILE`, `BLOB`, `CLOB`, and `NCLOB`) can provide many advantages over `LONG` data types.

In the following example, the `LONG` column named `long_col` in table `long_tab` is changed to data type `CLOB`:

```
SQL> 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 are still usable. However, all the indexes, including Domain indexes and Functional indexes, on all columns of the table become unusable and must be rebuilt using an `ALTER INDEX...REBUILD` statement. Also, the Domain indexes on the `LONG` column must be dropped before changing the `LONG` column to a LOB.

> **See Also:**
>
> *Oracle Database SecureFiles and Large Objects Developer's Guide* for information about modifying applications to use LOB data

Migrate Your Upgraded Oracle Databases to Use Unified Auditing

To use the full facilities of unified auditing, you must manually migrate to unified auditing.

In unified auditing, all Oracle Database audit trails (`SYS.AUD$` for the database audit trail, `SYS.FGA_LOG$` for fine-grained auditing, `DVYS.AUDIT_TRAIL$` for Database Vault, and so on) are combined into one single audit trail, which you can view by querying the `UNIFIED_AUDIT_TRAIL` data dictionary view for single-instance installations and `GV$UNIFIED_AUDIT_TRAIL` for Oracle Real Application Clusters environments.

- Understanding Unified Auditing Migration Process for Oracle Database
  Decide which audit policies you want to use in the upgraded database.
- Migrating to Unified Auditing for Oracle Database
  Use this procedure for multitenant container (CDB) databases to migrate to unified auditing.
- About Managing Earlier Audit Records After You Migrate to Unified Auditing
  Review, archive, and purge earlier audit trails in preparation for using the unified audit trail.
- Removing the Unified Auditing Functionality
  Use this procedure to remove unified auditing, and to use mixed-mode audit.

- Obtaining Documentation References if You Choose Not to Use Unified Auditing
  You can access documentation listed here to obtain configuration information about how to use non-unified auditing.

> **See Also:**
>
> *Oracle Database Security Guide* for information about how the audit features have changed for this release

Understanding Unified Auditing Migration Process for Oracle Database

Decide which audit policies you want to use in the upgraded database.

By default, unified auditing is not enabled for upgraded databases. If you have upgraded from an earlier release to Oracle Database 12*c*, then your database uses the same auditing functionality that was used in the earlier release. For newly created databases, the mixed-mode method of unified auditing is enabled by default. After you complete the migration to unified auditing, traditional auditing is disabled, and the new audit records write to the unified audit trail.

To enable and configure the audit policies and how they are used, choose one method as follows:

- Use the pure unified audit facility.

  Migrate to unified auditing to use the full unified auditing facility features. After you complete the procedure to migrate to unified auditing, you can create and enable new audit policies and also use the predefined audit policies. The audit records for these policies write to the unified audit trail. The earlier audit trails and their audit records remain, but no new audit records write to the earlier audit trails.

  > **Note:**
  >
  > The audit configuration from the earlier release has no effect in the unified audit system. Only unified audit policies generate audit records inside the unified audit trail.

- Use a mixed-mode audit facility.

  The mixed-mode audit facility enables both traditional and unified auditing facilities to run simultaneously and applies to both new and upgraded databases. The mixed-mode unified auditing facility becomes available if you enable at least one of the unified auditing predefined audit policies. Audit records for these policies write to the unified audit trail. The audit configuration in the earlier release of Oracle Database is also available, and the audit records for this configuration write to the earlier audit trails. If you decide that you prefer using the pure unified audit facility, then you can migrate to it.

> **Note:**
>
> If the database is not writable, then audit records write to new format operating system files in the `$ORACLE_BASE/audit/$ORACLE_SID` directory.

> **See Also:**
>
> - *Oracle Database Security Guide* for information about the predefined audit policies
> - *Oracle Database Security Guide* for information about the `ora_SecureConfig` audit policy

Migrating to Unified Auditing for Oracle Database

Use this procedure for multitenant container (CDB) databases to migrate to unified auditing.

In CDB environments, perform the following procedure in the `root`. The procedure migrates both the `root`CDB, and any associated PDBs, to unified auditing.

1. Log in to SQL*Plus as user `SYS` with the `SYSDBA` privilege.

   ```
   sqlplus sys as sysdba
   Enter password: password
   ```

   In a Pluggable Databases environment, this login connects you to `root`.

2. Check if your Oracle Database is migrated to unified auditing using this query:

   ```
   SQL> SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Unified
   Auditing';
   ```

   If the output for the `VALUE` column is `TRUE`, then unified auditing is already enabled in your database. You can proceed to Managing Earlier Audit Records. If the output is `FALSE`, then complete the remaining steps in this procedure.

3. Stop the database. For single-instance environments, enter the following commands from SQL*Plus:

   ```
   SQL> SHUTDOWN IMMEDIATE
   SQL> EXIT
   ```

   For Windows systems, stop the Oracle service:

   ```
   net stop OracleService%ORACLE_SID%
   ```

For Oracle RAC installations, shut down each database instance as follows:

```
srvctl stop database -db db_name
```

4. Stop the listener. (Stopping the listener is not necessary for Oracle RAC and Oracle Grid Infrastructure listeners.)

   ```
   lsnrctl stop listener_name
   ```

   You can find the name of the listener by running the `lsnrctl status` command. The `Alias` setting indicates the name.

5. Go to the directory `$ORACLE_HOME/rdbms/lib`.
6. Enable unified auditing for the Oracle user.
   - Linux and UNIX

     ```
     make -f ins_rdbms.mk uniaud_on ioracle ORACLE_HOME=$ORACLE_HOME
     ```

   - Windows

     Rename the file `%ORACLE_HOME%/bin/orauniaud12.dll.dbl` to `%ORACLE_HOME%/bin/orauniaud12.dll`.

   > **Note:**
   >
   > For Oracle RAC databases that have non-shared Oracle homes, you must repeat this step on each cluster member node, so that the binaries are updated inside the local ORACLE_HOME on each cluster node.

7. Restart the listener.

   ```
   lsnrctl start listener_name
   ```

8. Restart the database.

   Log in to SQL*Plus and then enter the `STARTUP` command:

   ```
   sqlplus sys as sysoper
   Enter password: password

   SQL> STARTUP
   ```

   For Windows systems, start the Oracle service:

   ```
   net start OracleService%ORACLE_SID%
   ```

   For Oracle RAC installations, start each database instance:

   ```
   srvctl start database -db db_name
   ```

About Managing Earlier Audit Records After You Migrate to Unified Auditing

Review, archive, and purge earlier audit trails in preparation for using the unified audit trail.

After you complete the procedure to migrate Oracle Database to use unified auditing, any audit records that your database had before remain in their earlier audit trails. You can archive these audit records and then purge their audit trails. With unified auditing in place, any new audit records write to the unified audit trail.

> **See Also:**
>
> - "Archiving the Audit Trail" in *Oracle Database Security Guide*
> - "Purging Audit Trail Records" in *Oracle Database Security Guide*

Removing the Unified Auditing Functionality

Use this procedure to remove unified auditing, and to use mixed-mode audit.

After you have enabled your databases to use unified auditing, if you decide that you do not want unified auditing, then you can use this procedure to remove the unified auditing functionality. In this case, your database uses the mixed-mode audit facility.

1. Stop the database.

   ```
   sqlplus sys as sysoper
   Enter password: password

   SQL> SHUTDOWN IMMEDIATE
   SQL> EXIT
   ```

   For Windows systems, stop the Oracle service:

   ```
   net stop OracleService%ORACLE_SID%
   ```

   For Oracle RAC installations, shut down each database instance as follows:

   ```
   srvctl stop database -db db_name
   ```

2. Go to the `$ORACLE_HOME/rdbms/lib` directory.
3. Disable the unified auditing executable.
   - **Unix:** Run the following command:

     ```
     make -f ins_rdbms.mk uniaud_off ioracle ORACLE_HOME=$ORACLE_HOME
     ```

   - **Microsoft Windows:** Rename the `%ORACLE_HOME%/bin/orauniaud12.dll` file to `%ORACLE_HOME%/bin/orauniaud12.dll.dbl`.

4. Restart the database.

```
sqlplus sys as sysoper
Enter password: password

SQL> STARTUP
SQL> EXIT
```

For Microsoft Windows systems, start the Oracle service again.

```
net start OracleService%ORACLE_SID%
```

For Oracle RAC installations, start each database instance using the following syntax:

```
srvctl start database -db db_name
```

Obtaining Documentation References if You Choose Not to Use Unified Auditing

You can access documentation listed here to obtain configuration information about how to use non-unified auditing.

After upgrading to the new release Oracle Database, if you choose not to change to unified auditing, then Oracle documentation and Oracle Technology Network provide information about traditional non-unified auditing.

- *Oracle Database Security Guide*: This guide is the main source of information for configuring auditing. You must use the Oracle Database Release 11g version of this manual. To access this guide:
  1. Visit the database page on `docs.oracle.com` site on Oracle Technology Network:

     `https://docs.oracle.com/en/database/index.html`
  2. Select **Oracle Database**.
  3. In the Downloads page, select the **Documentation** tab.
  4. On the release list field, select **Earlier Releases**, and select Oracle Database 11g Release 2 (11.2).
  5. From the Oracle Database 11g Release 2 (11.2) Documentation page, select the **All Books** link to display publications in the documentation set.
  6. Search for *Security Guide*.
  7. Select either the **HTML** or the **PDF** link for this guide.

Identify Oracle Text Indexes for Rebuilds

You can run a script that helps you to identify Oracle Text index indexes with token tables that can benefit by being rebuilt after upgrading to the new Oracle Database release..

When you upgrade from Oracle Database 12c release 1 (12.2.0.1) to Oracle Database 18c and later releases, the Oracle Text token tables ($I, $P, and so on) are expanded from 64 bytes to 255 bytes. However, if you have indexes with existing token tables using the smaller size range, then the Oracle Text indexes cannot take advantage of this widened token column range. You must rebuild the indexes to use the 255 byte size range. Oracle provides a script that can assist you to identify indexes that can benefit by being rebuilt.

Obtain the script from My Oracle Support:

https://support.oracle.com/rs?type=doc&id=2287094.1

Dropping and Recreating DBMS_SCHEDULER Jobs

If DBMS_SCHEDULER jobs do not function after upgrading from an earlier release, drop and recreate the jobs.

If you find that DBMS_SCHEDULER jobs are not functioning after an upgrade. drop and recreate those jobs. This issue can occur even if the upgrade process does not report issues, and system objects are valid.

Transfer Unified Audit Records After the Upgrade

Review these topics to understand how you can obtain better performance after you upgrade and migrate to unified auditing

- About Transferring Unified Audit Records After an Upgrade
  Transferring the unified audit records from Oracle Database 12c release 12.1 to the new relational table under the AUDSYS schema for the new Oracle Database release improves the read performance of the unified audit trail.
- Transferring Unified Audit Records After an Upgrade
  You can transfer unified audit records to the new relational table in AUDSYS by using the DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS PL/SQL procedure.

About Transferring Unified Audit Records After an Upgrade

Transferring the unified audit records from Oracle Database 12c release 12.1 to the new relational table under the AUDSYS schema for the new Oracle Database release improves the read performance of the unified audit trail.

Starting with Oracle Database 12c Release 2, unified audit records are written directly to a new internal relational table that is located in the AUDSYS schema. In Oracle Database 12c release 12.1, the unified audit records were written to the common logging infrastructure (CLI) SGA queues. If you migrated to unified auditing in that release, then to obtain better read performance, you can transfer the unified audit records that are from that release to the new Oracle Database release internal table. It is not mandatory that you perform this transfer, but Oracle recommends that you do so to obtain better unified audit trail read performance. This is a one-time operation. All new unified audit records that are generated after the upgrade are written to the new table. The table is a read-only table. Any attempt to modify the metadata or data of this table is mandatorily audited.

After you upgrade to the new Oracle Database release, if you have any unified audit records present in the UNIFIED_AUDIT_TRAIL from the earlier release, then consider

transferring them to the new internal relational table by using the transfer procedure for better read performance of the unified audit trail.

As with the SYS schema, you cannot query the AUDSYS schema if you have the SELECT ANY TABLE system privilege. In addition, this table is not listed as a schema object in the ALL_TABLES data dictionary view unless you have either the SELECT ANY DICTIONARY system privilege or an explicit SELECT privilege on this internal table. Until the database is open read write, the audit records are written to operating system spillover files (.bin format). However, you can transfer the audit records in these operating system files to the internal relational table after the database opens in the read write mode by using the DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES procedure.

Transferring Unified Audit Records After an Upgrade

You can transfer unified audit records to the new relational table in AUDSYS by using the DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS PL/SQL procedure.

1. Log in to the database instance as a user who has been granted the AUDIT_ADMIN role.

   For example, in a non-multitenant environment:

   ```
   sqlplus sec_admin
   Enter password: password
   ```

   For a multitenant environment, connect to the root:

   ```
   sqlplus c##sec_admin@root
   Enter password: password
   ```

   You can perform this procedure execution in the root as well as in a PDB, because the UNIFIED_AUDIT_TRAIL view is container specific. In addition, the transfer procedure is container specific. That is, performing the transfer from the root does not affect the unified audit records that are present in the unified audit trail for the PDB.

2. For a multitenant environment, query the DBA_PDB_HISTORY view to find the correct GUID that is associated with the CLI table that is specific to the container from which audit records must be transferred.

   For example:

   ```
   SQL> SELECT PDB_NAME, PDB_GUID FROM DBA_PDB_HISTORY;

   PDB_NAME  PDB_GUID
   --------  --------------------------------
   HR_PDB    33D96CA7862D53DFE0534DC0E40A7C9B
   ...
   ```

3. In a multitenant environment, connect to the container for which you want to transfer the audit records.

   You cannot perform the transfer operation on a container that is different from the one in which you are currently connected.

4. Run the DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS procedure.

   For example:

```
SQL> EXEC DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS;

PL/SQL procedure successfully completed.
```

Or, to specify the PDB GUID:

```
SQL> EXEC DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS
('33D96CA7862D53DFE0534DC0E40A7C9B');

PL/SQL procedure successfully completed.
```

5. If the database is in open read write mode, then execute the `DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES` procedure.

   Until the database is in open read write mode, audit records are written to operating system (OS) files. The `DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES` procedure moves the unified audit records that are present in the files to database tables. You can find the unified audit records that are present in the OS spillover files by querying the `V$UNIFIED_AUDIT_TRAIL` dynamic view.

   For example, if you want to execute this procedure for audit records in the `HR_PDB` container, then you must connect to that PDB first:

   ```
   SQL> CONNECT sec_admin@HR_PDB
   Enter password: password

   SQL> EXEC DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES;

   PL/SQL procedure successfully completed.
   ```

6. Query the `UNIFIED_AUDIT_TRAIL` data dictionary view to check if the records transferred correctly.

   Oracle highly recommends that you query `UNIFIED_AUDIT_TRAIL`. After a successful audit record transfer, you should query the `UNIFIED_AUDIT_TRAIL` because querying the `V$UNIFIED_AUDIT_TRAIL` dynamic view will show the audit records that are present only in the OS spillover files.

About Testing the Upgraded Production Oracle Database

Repeat tests on your production database that you carried out on your test database to ensure applications operate as expected.

If you upgraded a test database to the new Oracle Database release, and then tested it, then you can now repeat those tests on the production database that you upgraded to the new Oracle Database release. Compare the results, noting anomalies. Repeat the test upgrade as many times as necessary.

To verify that your applications operate properly with a new Oracle Database release, test the newly upgraded production database with your existing applications. You also can test enhanced functions by adding available Oracle Database features, and then testing them. However, first ensure that the applications operate in the same manner as they did before the upgrade.

Upgrading the Time Zone File Version After Upgrading Oracle Database

If the Pre-Upgrade Information Tool instructs you to upgrade the time zone files after completing the database upgrade, then use the `DBMS_DST` PL/SQL package to upgrade the time zone file.

Oracle Database supplies multiple versions of time zone files. There are two types of files associated with each time zone file: a large file, which contains all the time zones defined in the database, and a small file, which contains only the most commonly used time zones. The large versions are designated as `timezlrg_`*`version_number`*`.dat`. The small versions are designated as `timezone_`*`version_number`*`.dat`. The files are located in the `oracore/zoneinfo` subdirectory under the Oracle Database home directory.

Related Topics

- *Oracle Database Globalization Support Guide*
- https://support.oracle.com/rs?type=doc&id=1585343.1

Recommended Tasks After Upgrading an Oracle RAC Database

Decide if you want to configure clients to use SCAN or node listeners for connections.

Oracle Real Application Clusters 12*c* uses the Single Client Access Name (SCAN). The SCAN is a single name that resolves to three IP addresses in the public network. When you upgrade a release of an Oracle RAC database earlier than release 11.2, the Oracle RAC database is registered with SCAN listeners as remote listeners. The Oracle RAC database also continues to register with all node listeners. SCAN listeners offer a variety of benefits. These benefits include enabling you to configure clients one time, and adding or removing nodes from the cluster without needing to change client connection configurations.

You can configure clients to use SCANs, or you can continue to use listeners configured on cluster member nodes. If you migrate all of your client connections to use SCANs, then you can remove the node listeners from the `REMOTE_LISTENERS` parameter. However, you cannot remove the node listeners themselves, because only node listeners can create dedicated servers for the database.

> **See Also:**
>
> *Oracle Clusterware Administration and Deployment Guide* for more information about Single Client Access Names (SCAN)

Recommended Tasks After Upgrading Oracle ASM

After you have upgraded Oracle ASM, Oracle recommends that you perform tasks such as resetting the Oracle ASM passwords and configuring disk groups.

- Create a Shared Password File In the ASM Diskgroup
  If you advance the `COMPATIBLE.ASM` disk group attribute, then create a shared password file.
- Reset Oracle ASM Passwords to Enforce Case-Sensitivity
  To take advantage of enforced case-sensitive passwords, you must reset the passwords of existing users during the database upgrade procedure.
- Advancing the Oracle ASM and Oracle Database Disk Group Compatibility
  You can advance the Oracle Database and the Oracle ASM disk group compatibility settings across software versions.
- Set Up Oracle ASM Preferred Read Failure Groups
  Oracle ASM administrators can specify some disks as preferred read disks for read I/O operations.

Related Topics

- Add New Features as Appropriate
  Review new features as part of your database upgrade plan.
- Develop New Administrative Procedures as Needed
  Plan a review of your scripts and procedures, and change as needed.

Create a Shared Password File In the ASM Diskgroup

If you advance the `COMPATIBLE.ASM` disk group attribute, then create a shared password file.

If you advanced the `COMPATIBLE.ASM` disk group attribute to 12.1 or later, then you are required to create a shared password file in the ASM diskgroup.

> **See Also:**
>
> *Oracle Automatic Storage Management Administrator's Guide* for complete information about managing a shared password file in a disk group

Reset Oracle ASM Passwords to Enforce Case-Sensitivity

To take advantage of enforced case-sensitive passwords, you must reset the passwords of existing users during the database upgrade procedure.

In releases earlier than Oracle Database 11*g* Release 1 (11.1), passwords are not case sensitive. You can enforce case sensitivity for passwords. For example, the password `hPP5620qr` fails if it is entered as `hpp5620QR` or `hPp5620Qr`.

For new Oracle ASM instances, there are no additional tasks or management requirements. For upgraded Oracle ASM instances, each user password must be reset with an `ALTER USER` statement.

> **Note:**
>
> If the default Oracle Database security settings are in place, then passwords must be at least eight characters, and passwords such as `welcome` and `oracle` are not allowed. See *Oracle Database Security Guide* for more information.

Advancing the Oracle ASM and Oracle Database Disk Group Compatibility

You can advance the Oracle Database and the Oracle ASM disk group compatibility settings across software versions.

> **Caution:**
>
> If you advance the `COMPATIBLE.RDBMS` attribute, then you cannot revert to the previous setting. Before advancing the `COMPATIBLE.RDBMS` attribute, ensure that the values for the `COMPATIBLE` initialization parameter for all of the databases that use the disk group are set to at least the new setting for `COMPATIBLE.RDBMS` before you advance the attribute value.

Advancing compatibility enables new features only available in the new release. However, doing so makes the disk group incompatible with older releases of the software. Advancing the on disk compatibility is an irreversible operation.

Use the `compatible.rdbms` and `compatible.asm` attributes to specify the minimum software release required by the database instance and the Oracle ASM instance, respectively, to access the disk group. For example, the following `ALTER DISKGROUP` statement advances the Oracle ASM compatibility of the disk group `asmdg2`:

```
ALTER DISKGROUP asmdg2 SET ATTRIBUTE 'compatible.asm' = '12.2'
```

In this case, the disk group can be managed only by Oracle ASM software of release 12.2 or later, while any database client of release 11.2 or later can use the disk group.

Set Up Oracle ASM Preferred Read Failure Groups

Oracle ASM administrators can specify some disks as preferred read disks for read I/O operations.

When an ASM administrator defines Oracle ASM preferred read failure groups, Oracle ASM can then read from the extent that is in the nearest preferred read disk, rather than always reading the primary copy.

See Also:

- *Oracle Clusterware Administration and Deployment Guide* for information about specifying failure groups settings in an extended cluster
- *Oracle Automatic Storage Management Administrator's Guide* for complete information about Oracle ASM preferred read failure groups, and specifying the new `ASM_PREFERRED_READ_FAILURE_GROUPS` initialization parameter to list failure group names that contain the preferred read disks for each node in a cluster
- *Oracle Database Reference* for the `ASM_PREFERRED_READ_FAILURE_GROUPS` initialization parameter

Recommended Tasks After Upgrading Oracle Database Express Edition

Use DBCA or run manual scripts to install additional components into Oracle Database.

An Oracle Database Express database contains only a subset of the components available in an Oracle Database Standard Edition or Oracle Database Enterprise Edition database. After upgrading to the new Oracle Database release, you can use Database Configuration Assistant (DBCA) or manual scripts to install additional components into your database.

Tasks to Complete Only After Manually Upgrading Oracle Database

After you complete your upgrade, you must perform the tasks described here if you upgrade your database manually instead of using DBUA.

- Changing Passwords for Oracle Supplied Accounts
  Oracle recommends that you carry out these tasks to protect new Oracle user accounts.
- Create or Migrate Your Password File with ORAPWD
  Review if you have `REMOTE_LOGIN_PASSWORDFILE` set.
- Migrating Your Initialization Parameter File to a Server Parameter File
  If you are currently using a traditional initialization parameter file, then use this procedure to migrate to a server parameter file.
- Identifying and Copying Oracle Text Files to a New Oracle Home
  To upgrade Oracle Text, use this procedure to identify and copy required files from your existing Oracle home to the new release Oracle home. Complete this task after you upgrade Oracle Database.
- Upgrading the Oracle Clusterware Configuration
  If you are using Oracle Clusterware, then you must upgrade the Oracle Clusterware keys for the database.

- Adjust the Initialization Parameter File for the New Release
  Review these topics to help you to check your initialization parameters after upgrading.
- Set CLUSTER_DATABASE Initialization Parameter For Oracle RAC After Upgrade
  For manual upgrades of Oracle RAC database instances, you must change the CLUSTER_DATABASE initialization parameter to rejoin the node to the new release cluster.

Changing Passwords for Oracle Supplied Accounts

Oracle recommends that you carry out these tasks to protect new Oracle user accounts.

Depending on the release from which you upgraded, there may be new Oracle user accounts on your database. Oracle recommends that you lock all Oracle supplied accounts except for `SYS` and `SYSTEM`, and expire their passwords, so that new passwords are required when the accounts are unlocked.

> **Note:**
>
> If the default Oracle Database 12*c* security settings are in place, then passwords must be at least eight characters, and passwords such as `welcome` and `oracle` are not allowed.

> **See Also:**
>
> *Oracle Database Security Guide* about password requirements

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:

```
SQL> ALTER USER username PASSWORD EXPIRE ACCOUNT LOCK;
```

Create or Migrate Your Password File with ORAPWD

Review if you have `REMOTE_LOGIN_PASSWORDFILE` set.

If the `REMOTE_LOGIN_PASSWORDFILE` initialization parameter is set to `EXCLUSIVE`, then create or migrate the password file with `ORAPWD`. Oracle Database 12*c* and later releases provide a new option to `ORAPWD` for migrating the password file from your existing database.

With Oracle Database 12c release 2 (12.2) and later releases, if `REMOTE_LOGIN_PASSWORDFILE` is set to `SHARED`, then you receive a pre-upgrade

check validation warning. You can choose one of the following options to correct this issue:

- Disable the password file-based authentication entirely by setting `REMOTE_LOGIN_PASSWORDFILE = NONE`
- Limit the password file-based authentication by setting `REMOTE_LOGIN_PASSWORDFILE = EXCLUSIVE`

> **See Also:**
>
> *Oracle Database Administrator's Guide* for more information about creating or migrating password files

Migrating Your Initialization Parameter File to a Server Parameter File

If you are currently using a traditional initialization parameter file, then use this procedure to migrate to a server parameter file.

1. If the initialization parameter file is located on a client computer, then transfer the file from the client computer to the server computer.
2. Create a server parameter file using the `CREATE SPFILE` statement. This statement reads the initialization parameter file to create a server parameter file. You are not required to start the database to issue a `CREATE SPFILE` statement.
3. Start up the instance using the newly-created server parameter file.

> **Note:**
>
> If you are using Oracle RAC, then you must combine all of your instance-specific initialization parameter files into a single initialization parameter file. Complete the procedures necessary for using a server parameter file with cluster databases.

Related Topics

- *Oracle Real Application Clusters Administration and Deployment Guide*

Identifying and Copying Oracle Text Files to a New Oracle Home

To upgrade Oracle Text, use this procedure to identify and copy required files from your existing Oracle home to the new release Oracle home. Complete this task after you upgrade Oracle Database.

Certain Oracle Text features rely on files under the Oracle home that you have configured. After manually upgrading to a new Oracle Database release, or after any process that changes the Oracle home, you must identify and move these files manually. These files include user filters, mail filter configuration files, and all knowledge base extension files. After you identify the files, copy the files from your existing Oracle home to the new Oracle home.

To identify and copy required files from your existing Oracle home to the new release Oracle home:

1. Log in with the `SYS`, `SYSTEM`, or `CTXSYS` system privileges for the upgraded database.
2. Under the Oracle home of the upgraded database, run the `$ORACLE_HOME/ctx/admin/ctx_oh_files.sql` SQL script.

   For example:

   ```
   sqlplus / as sysdba
   connected
   SQL> @?/ctx/admin/ctx_oh_files
   ```

3. Review the output of the `ctx_oh_files.sql` command, and copy the files to the new Oracle home.

Related Topics

- *Oracle Text Application Developer's Guide*

Upgrading the Oracle Clusterware Configuration

If you are using Oracle Clusterware, then you must upgrade the Oracle Clusterware keys for the database.

Run `srvctl` for Oracle Database 12*c* to upgrade the database. For example:

```
ORACLE_HOME/bin/srvctl upgrade database -db name -o ORACLE_HOME
```

Related Topics

- *Oracle Real Application Clusters Administration and Deployment Guide*

Adjust the Initialization Parameter File for the New Release

Review these topics to help you to check your initialization parameters after upgrading.

Each release of Oracle Database introduces new initialization parameters, deprecates some initialization parameters, and desupports some initialization parameters. You must adjust the parameter file to account for these changes, and to take advantage of new initialization parameters that can be beneficial to your system. Additionally, when you perform a manual upgrade without using DBUA, the `tnsnames.ora` file is not automatically populated with new configuration information and settings. Therefore, you must manually update `tnsnames.ora` and adjust `local_listener` and `remote_listener` parameter references if these must be resolved.

- Setting the COMPATIBLE Initialization Parameter After Upgrade
  After testing, you can set the `COMPATIBLE` initialization parameter to the compatibility level you want for your new database.
- Adjust TNSNAMES.ORA and LISTENER Parameters After Upgrade
  After performing a manual upgrade, if you must resolve `local_listener` and `remote_listener` in `tnsnames.ora`, then you must manually adjust those parameters.

See Also:

- *Oracle Database Reference* “Changes In this Release” section for a list of new initialization parameters, and for information about each parameter

Setting the COMPATIBLE Initialization Parameter After Upgrade

After testing, you can set the `COMPATIBLE` initialization parameter to the compatibility level you want for your new database.

The `COMPATIBLE` initialization parameter controls the compatibility level of your database. Set the `COMPATIBLE` initialization parameter to a higher value only when you are certain that you no longer need the ability to downgrade your database.

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

   Raising the `COMPATIBLE` initialization parameter can cause your database to become incompatible with earlier releases of Oracle Database. A backup ensures that you can return to the earlier release if necessary.

2. If you are using a server parameter file, then complete the following steps:

   a. To set or change the value of the `COMPATIBLE` initialization parameter, update the server parameter file.

      For example, to set the `COMPATIBLE` initialization parameter to `12.2.0`, enter the following statement:

      ```
      SQL> ALTER SYSTEM SET COMPATIBLE = '12.2.0' SCOPE=SPFILE;
      ```

   b. Shut down and restart the instance.

3. If you are using an initialization parameter file, then complete the following steps:

   a. If an instance is running, then shut it down.

      For example:

      ```
      SQL> SHUTDOWN IMMEDIATE
      ```

   b. To set or change the value of the `COMPATIBLE` initialization parameter, you edit the initialization parameter file.

      For example, to set the `COMPATIBLE` initialization parameter to `for Oracle Database release 12.2`, enter the following in the initialization parameter file:

      ```
      COMPATIBLE = 12.2.0
      ```

   c. Start the instance using `STARTUP`.

> **Note:**
>
> If you are using an ASM disk group, then the disk group compatibility attribute must be equal to or less than the value for the database compatibility parameter in `init.ora`.

> **See Also:**
>
> *Oracle Database Backup and Recovery User's Guide* for more information about performing a backup

Adjust TNSNAMES.ORA and LISTENER Parameters After Upgrade

After performing a manual upgrade, if you must resolve `local_listener` and `remote_listener` in `tnsnames.ora`, then you must manually adjust those parameters.

DBUA handles changes to network naming and listeners during automatic upgrades. However, during a manual upgrade, neither `tnsnames.ora` nor the listeners are changed.

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database Net Services Reference*

Set CLUSTER_DATABASE Initialization Parameter For Oracle RAC After Upgrade

For manual upgrades of Oracle RAC database instances, you must change the CLUSTER_DATABASE initialization parameter to rejoin the node to the new release cluster.

In upgrades of cluster member nodes, you set the `CLUSTER_DATABASE` initialization parameter to `false` before upgrading a cluster database.

After you complete the upgrade, you must set this parameter to `true`, so that you can rejoin the node to the new release cluster.

> **Note:**
>
> If you carry out your upgrade using Database Upgrade Assistant (DBUA), then DBUA performs this task for you.

7

Upgrading Applications After Upgrading Oracle Database

To take full advantage of new features, you must upgrade applications running in the new release.

Many new features and enhancements are available after upgrading to a new release of Oracle Database. Review these topics for guidance in planning these application upgrades.

- Overview of Upgrading Applications on a New Oracle Database Release
  You are not required to modify existing applications that do not use features available in the new Oracle Database release.
- Compatibility Issues for Applications on Different Releases of Oracle Database
  You can encounter compatibility issues between different releases of Oracle Database that can affect your applications.
- Software Upgrades and Client and Server Configurations for Oracle Database
  Use these topics to understand your options for upgrading precompiler and Oracle Call Interface (OCI) applications, depending on the type of software upgrade that you are performing and your client and server configurations.
- Compatibility Rules for Applications When Upgrading Oracle Database Client or Server Software
  Compatibility rules apply when you upgrade Oracle Database client or server software.
- About Upgrading Precompiler and OCI Applications in Oracle Database
  Review this information if you want to upgrade precompiler and Oracle Call Interface (OCI) applications.
- Schema-Only Accounts and Upgrading EXPIRED Password Accounts
  Before starting your upgrade, determine if you want to use password authentication to default Oracle Database accounts where their passwords are in `EXPIRED` status, and their account is in `LOCKED` status
- About Upgrading Options for Oracle Precompiler and OCI Applications
  Oracle provides several options for upgrading your precompiler and Oracle Call Interface (OCI) applications running on a new release of Oracle Database.
- Upgrading SQL*Plus Scripts and PL/SQL after Upgrading Oracle Database
  To use features and functions of the new Oracle Database release, you must change existing SQL scripts to use the syntax of the new Oracle Database release.
- About Upgrading Oracle Forms or Oracle Developer Applications
  Review Oracle Forms and Oracle Developer new features to see if any of your applications can benefit from them.

Schema-Only Accounts and Upgrading EXPIRED Password Accounts

Before starting your upgrade, determine if you want to use password authentication to default Oracle Database accounts where their passwords are in `EXPIRED` status, and their account is in `LOCKED` status

During upgrades to Oracle Database 19c and later releases, default Oracle accounts that have not had their passwords reset before upgrade (and are set to `EXPIRED` status), and that are also set to `LOCKED` status, are set to `NO AUTHENTICATION` after the upgrade is complete.

Because of this new feature, default accounts that are changed to schema-only accounts become unavailable for password authentication. The benefit of this feature is that administrators no longer have to periodically rotate the passwords for these Oracle Database-provided schemas. This feature also reduces the security risk of attackers using default passwords to hack into these accounts.

If you want to prevent these Oracle accounts from being set to schema-only accounts during the upgrade, then you must either set a valid strong password for the account before you start the upgrade, or set a valid strong password for these accounts after upgrade, or unlock the accounts before you log in to the upgraded Oracle Database.

After the upgrade, an administrator can also enable password authentication for schema-only accounts. However, for better security, Oracle recommends that you keep these accounts as schema only accounts.

Related Topics

- *Oracle Database Security Guide*

Overview of Upgrading Applications on a New Oracle Database Release

You are not required to modify existing applications that do not use features available in the new Oracle Database release.

Existing applications running in a new release of Oracle Database function the same as they did in earlier releases, and achieve the same, or enhanced, performance.

Many new features and enhancements are available after upgrading to the new Oracle Database release. Some of these changes provide added features and functions, while others provide improved performance. Before you upgrade your applications, Oracle recommends that you review and fully test these new features to decide which ones you want to use.

Related Topics

- Database Features and Licensing App

Compatibility Issues for Applications on Different Releases of Oracle Database

You can encounter compatibility issues between different releases of Oracle Database that can affect your applications.

Compatibility issues can occur due to differences between Oracle Database releases. Also, in each new release of Oracle Database, new Oracle reserved words can be added, or initialization parameters can be changed, or the data dictionary can be changed. Review the relevant topics in this documentation for more information.

When you upgrade your Oracle Database software to a new release, ensure that your applications do not use any words reserved by Oracle, that your applications are compatible with the initialization parameters of the database, and that your applications are compatible with the data dictionary of the database.

Also be aware that new releases of Oracle Database can be supported only on particular operating system releases or patch sets. An operating system release and patch set that is supported for use with a previous release of Oracle Database can not be supported for current releases. Check operating system requirements before you install oracle software to perform an upgrade. In addition, to be able to use some features, your system can require additional patch sets or kernel additions.

Related Topics

- Oracle SQL Reserved Words and Keywords

Software Upgrades and Client and Server Configurations for Oracle Database

Use these topics to understand your options for upgrading precompiler and Oracle Call Interface (OCI) applications, depending on the type of software upgrade that you are performing and your client and server configurations.

- Possible Client and Server Configurations for Oracle Database
  Select a client/server configuration to run your precompiler and OCI applications.
- Types of Software Upgrades for Oracle Database Client and Server Software
  Review to identify if a release is a major release or a maintenance release.

Possible Client and Server Configurations for Oracle Database

Select a client/server configuration to run your precompiler and OCI applications.

Your precompiler and OCI applications run on the client in a client/server environment, where the Oracle Database server is the server. You can use one or more of the following client/server configurations in your environment

Oracle Database Client and Server on 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.

Oracle Database Client and Server in Different Oracle Locations 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.

Oracle Database Client and Server in the Same Oracle Location

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:**
>
> *Oracle Database Concepts* for more information about client/server environments

Types of Software Upgrades for Oracle Database Client and Server Software

Review to identify if a release is a major release or a maintenance release.

Two types of upgrades are possible for Oracle Database client and server software: a major release of Oracle Database, and a maintenance release for Oracle Database.

Oracle Database Major Release Upgrade

In a major release, the upgrade changes the first digit of the release number. For example, upgrading from Oracle Database 11*g* to Oracle Database 12*c* is a major database release upgrade.

Oracle Database Maintenance Release Upgrade

In a maintenance release, the upgrade changes the second digit of the release number. For example, upgrading from Oracle Database 12*c* Release 1 (12.1) to Oracle Database 12*g* release 2 (12.2) is a database maintenance release upgrade. However, Oracle can introduce new features in maintenance releases.

Compatibility Rules for Applications When Upgrading Oracle Database Client or Server Software

Compatibility rules apply when you upgrade Oracle Database client or server software.

Compatibility rules are based on the type of software upgrade you are performing, and the type of client/server configuration.

> **Note:**
>
> This section uses the terms introduced in “Software Upgrades and Client and Server Configurations.” .

- Rules for Upgrading Oracle Database Server Software
  Different rules apply when you upgrade Oracle Database server software depending on your database environment.
- Upgrading the Oracle Database Client Software
  Keeping the server and client software at the same release number ensures the maximum stability for your applications.

Rules for Upgrading Oracle Database Server Software

Different rules apply when you upgrade Oracle Database server software depending on your database environment.

- If You Do Not Change the Client Environment, Then You Are Not Required to Relink
  Review these scenarios to determine if you must relink your applications after upgrading.
- Applications Can Run Against Newer or Older Oracle Database Server Releases
  If you run a precompiler or OCI application against a database server, then Oracle recommends that the release of the database server software is equal to or later than the client software release.

If You Do Not Change the Client Environment, Then You Are Not Required to Relink

Review these scenarios to determine if you must relink your applications after upgrading.

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 Database server software without changing the client software, then you are not required to precompile, compile, or relink your applications.

In this set of scenarios, client software using Oracle Databases are in separate locations from the server software, and the client software continues to function without direct effects from the upgrade.

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 for upgrading Oracle Database client software.

> **Note:**
>
> You can upgrade the Oracle Database server software, but not install the new precompiler or OCI client software, when you are using the same Oracle home for both binaries. In this case, the client software is not upgraded. However, Oracle does not recommend this configuration.

Applications Can Run Against Newer or Older Oracle Database Server Releases

If you run a precompiler or OCI application against a database server, then Oracle recommends that the release of the database server software is equal to or later than the client software release.

This recommendation configuration is not strictly required.

For example: If your client software is Oracle 12*c* release 2 (12.2.0.1), then if you run precompiler applications on the client against, the server, Oracle recommends that your server software is Oracle 12*c* release 2 (12.2) or later.

Upgrading the Oracle Database Client Software

Keeping the server and client software at the same release number ensures the maximum stability for your applications.

Use this information to plan your Oracle Database Client installations. Depending on how your applications are linked, different rules apply when you upgrade the Oracle Database client software.

Oracle recommends that you upgrade your client software to match the current server software. For example, when you upgrade Oracle Database to the new Oracle Database release, Oracle recommends that you also upgrade your Oracle Database client software to the new release. The latest Oracle Database client software can provide added features and performance enhancements that are only available with that later release.

- About Image-Based Oracle Database Client Installation
  Starting with Oracle Database 19c, installation and configuration of Oracle Database Client software is simplified with image-based installation.
- About Linking Applications with Newer Libraries
  You can link the code generated by precompiler applications and Oracle Call Interface (OCI) with a release of the client library that equals or is later than the server release.
- Statically Linked Applications Must Always Be Relinked
  Statically-linked code can be incompatible with error messages in the upgraded ORACLE_HOME.
- About Relinking Dynamically Linked Applications
  Dynamically linked OCI applications from Oracle Database 10*g* Release 1 (10.1) and later releases are upward-compatible with the current release.

About Image-Based Oracle Database Client Installation

Starting with Oracle Database 19c, installation and configuration of Oracle Database Client software is simplified with image-based installation.

To install Oracle Database Client, create the new Oracle home, extract the image file into the newly-created Oracle home, and run the setup wizard to register the Oracle Database product.

You must extract the image software (`client_home.zip`) into the directory where you want your Oracle Database Client home to be located, and then run the Setup Wizard to start the Oracle Database Client installation and configuration. Oracle recommends that the Oracle home directory path you create is in compliance with the Oracle Optimal Flexible Architecture recommendations.

Using image-based installation, you can install Oracle Database Client 32-bit and 64-bit configurations of the Administrator installation type.

As with Oracle Database and Oracle Grid Infrastructure image file installations, Oracle Database Client image installations simplify Oracle Database Client installations and ensure best practice deployments. Oracle Database Client installation binaries continue to be available in the traditional format as non-image zip files.

About Linking Applications with Newer Libraries

You can link the code generated by precompiler applications and Oracle Call Interface (OCI) with a release of the client library that equals or is later than the server release.

The OCI runtime library that you use must either be the same release, or a later release, than the release of the OCI library with which the application was developed.

Statically Linked Applications Must Always Be Relinked

Statically-linked code can be incompatible with error messages in the upgraded ORACLE_HOME.

You must relink statically-linked OCI applications for both major and minor releases. The statically-linked Oracle client-side library code may be incompatible with the error messages in the upgraded ORACLE_HOME. For example, if an error message is updated with additional parameters, then it becomes incompatible with the statically-linked code.

About Relinking Dynamically Linked Applications

Dynamically linked OCI applications from Oracle Database 10*g* Release 1 (10.1) and later releases are upward-compatible with the current release.

The Oracle client-side dynamic library is upward-compatible with the previous version of the library. Oracle Universal Installer creates a symbolic link for the previous version of the library that resolves to the current version. Therefore, an application that is dynamically linked with the previous version of the Oracle client-side dynamic library does not require relinking to operate with the current version of the Oracle client-side library.

> **Note:**
>
> If the application is linked with a run-time library search path (such as `-rpath` on Linux), then the application may still run with the version of the Oracle client-side library with which it is linked. You must relink the application to run with the current version of the Oracle client-side library.
>
> If the application is linked with the deferred option (for example, statically-linked application), then it must be relinked.
>
> If the application is from a release earlier than Oracle Database 10*g* Release 1 (10.1), then it must be relinked.

About Upgrading Precompiler and OCI Applications in Oracle Database

Review this information if you want to upgrade precompiler and Oracle Call Interface (OCI) applications.

Testing precompiler and Oracle Call Interface upgrades consists of the following steps:

1. Create a test environment before you upgrade your production environment.
2. Include your upgraded application and the new Oracle Database software in your test environment.
3. Ensure that your test environment provides a realistic test of your application.

Related Topics

- Testing the Upgrade Process for Oracle Database

> **See Also:**
>
> - *Pro*C/C++ Programmer's Guide*
> - *Pro*COBOL Programmer's Guide*
> - *Oracle Call Interface Programmer's Guide*

About Upgrading Options for Oracle Precompiler and OCI Applications

Oracle provides several options for upgrading your precompiler and Oracle Call Interface (OCI) applications running on a new release of Oracle Database.

The upgrade 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 3 is the most difficult option, but it offers the most potential benefits.

- Option 1: Leave the Application Unchanged
  Leave the application and its environment unchanged.
- Option 2: Precompile or Compile the Application Using the New Software
  Application code must be changed if any APIs are deprecated or changed.
- Option 3: Change the Application Code to Use New Oracle Database Features
  Make code changes to your applications to take advantage of new Oracle Database features.
- Changing Oracle Precompiler and OCI Application Development Environments
  When you have decided on the new features to use, change the code of your application to use these features.

Option 1: Leave the Application Unchanged

Leave the application and its environment unchanged.

Do not relink, precompile, or compile the application, and do not change the application code. The application continues to work against the new Oracle Database release. This option requires that the Oracle home environment of the application is not upgraded. You can leave the application unchanged, and it continues to work with the new release Oracle Database server. 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 are not required to upgrade any 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 release of Oracle Database. In addition, your application cannot leverage all the possible performance benefits of the new Oracle Database release.

Option 2: Precompile or Compile the Application Using the New Software

Application code must be changed if any APIs are deprecated or changed.

Precompile or compile, and then relink the application using the new release of Oracle Database. When upgrading to the new release of Oracle Database software, you must precompile or compile the application with the new software after making necessary code changes to account for APIs that are deprecated or changed.

This option requires that you install the new Oracle Database client software on each client computer. You are required to precompile or compile, and relink your application only one time, regardless of the number of clients you have.

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 can emerge when you precompile or compile with the new Oracle Database software. Precompiling and compiling with the new software helps you detect and correct problems in the application code that previously were unnoticed.

Also, recompiling affords maximum stability for your application, because you are sure that it works with the new Oracle Database release. Further, your environment is ready for new development using the latest tools and features available. In addition, you

might benefit from performance improvements that are available with the new Oracle software only after you recompile and relink.

Option 3: Change the Application Code to Use New Oracle Database Features

Make code changes to your applications to take advantage of new Oracle Database features.

Change the application code to use new features in the new Oracle Database release. Then, precompile or compile and then relink the code. This option is the most difficult, but it can provide the most potential benefits. You gain all of the advantages described in Option 2: Precompile or Compile the Application Using the New Software. In addition, you also benefit from changes to your application that can leverage performance and scalability benefits available with the new release of Oracle Database. You can also add new features to your application that are available only with the new release. Consult the Oracle documentation for your development environment so that you understand how to implement the features thaqt you want to use.

> **See Also:**
>
> *Oracle Database New Features Guide* to become familiar with the features in this new Oracle Database release

Changing Oracle Precompiler and OCI Application Development Environments

When you have decided on the new features 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
  Complete these steps to change precompiler applications to use new Oracle Database release features.
- Changing OCI Applications
  To use new features in your new Oracle Database release, you must recompile your applications with the OCI calls for the new Oracle Database release.

Changing Precompiler Applications

Complete these steps to change precompiler applications to use new Oracle Database release features.

To use new features in a new Oracle Database release, you must add new code into your existing applications, and recompile the applications.

1. Incorporate the code for new features into your existing applications.

2. Precompile each application using the Oracle precompiler.
3. Compile each application.
4. Relink each application with the runtime library of the new Oracle Database release, `SQLLIB`, which is included with the precompiler.

Changing OCI Applications

To use new features in your new Oracle Database release, you must recompile your applications with the OCI calls for the new Oracle Database release.

1. Incorporate OCI calls of the new Oracle Database release into the existing application
2. Compile the application.
3. Relink the application with the new Oracle Database release runtime library.

Upgrading SQL*Plus Scripts and PL/SQL after Upgrading Oracle Database

To use features and functions of the new Oracle Database release, you must change existing SQL scripts to use the syntax of the new Oracle Database release.

If existing SQL scripts do not use features and functions of the new Oracle Database release, then they run unchanged on the new Oracle Database release, and require no modification.

Be aware that because of improved error checking in the new Oracle Database release, it may identify errors at compile time rather than at run time.

About Upgrading Oracle Forms or Oracle Developer Applications

Review Oracle Forms and Oracle Developer new features to see if any of your applications can benefit from them.

In Oracle Database 12c, *Oracle Database Development Guide* was renamed to *Oracle Database Advanced Application Developer's Guide*. Review that publication and the new features guide for information about changes to procedures for developing applications, and for new features of this Oracle Database release that affect application development.

8

Downgrading Oracle Database to an Earlier Release

For supported releases of Oracle Database, you can downgrade a database to the release from which you last upgraded.

For example, if you recently upgraded your Oracle Database 12c release 1 (12.1.0.2(to release 19c, and you did not change the compatible initialization parameter to 19.1.0, then you can downgrade to release 12.1.0.2.

- Supported Releases for Downgrading Oracle Database
  You can downgrade both major releases and release update or patchset releases, based on the original Oracle Database release from which the database was upgraded.
- Check for Incompatibilities When Downgrading Oracle Database
  To see if the database has incompatibilities that can prevent you from downgrading, check the compatibility level of your database.
- Perform a Full Backup Before Downgrading Oracle Database
  Oracle strongly recommends that you perform a full backup of your new Oracle Database release before you downgrade to a supported earlier release.
- Performing Required Predowngrade Steps for Oracle Database
  Complete the required preparation steps described here before you downgrade Oracle Database to the earlier release from which you upgraded.
- Using Scripts to Downgrade a CDB or Non-CDB Oracle Database
  To automate downgrades, starting in Oracle Database 19c, Oracle provides the `dbdowngrade` utility script. When necessary, you can also continue to run `catdwgrd.sql` manually, as in previous releases.
- About Downgrades and Invalid Objects with Component Status OPTION OFF
  Downgrades from later release Oracle Database Standard Edition to Oracle Database 12c Standard Edition with CDB and PDBS contain later release invalid objects after the downgrade.
- Downgrading a Single Pluggable Oracle Database (PDB)
  If you are downgrading Oracle Database, then you can downgrade one PDB without downgrading the whole CDB.
- Downgrading PDBs That Contain Oracle Application Express
  Use this procedure to avoid INVALID OBJECTS OWNED BY APEX_050000 errors when you downgrade PDBs that contain Oracle Application Express.
- Post-Downgrade Tasks for Oracle Database Downgrades
  After you downgrade your Oracle Database release, you can be required to complete additional tasks, due to changes that affect compatibility, components, and supported protocols.
- Troubleshooting the Downgrade of Oracle Database
  Use this troubleshooting information to address issues that may occur when downgrading Oracle Database.

Supported Releases for Downgrading Oracle Database

You can downgrade both major releases and release update or patchset releases, based on the original Oracle Database release from which the database was upgraded.

Releases Supported for Downgrades

You can downgrade a non-CDB Oracle Database from Oracle Database 19c to Oracle Database 18c, Oracle Database 12c, and to Oracle Database 11g release 11.2.0.4.

You can downgrade a PDB or CDB to Oracle Database 18c, Oracle Database 12c Release 12.2, and Oracle Database 12c Release 12.1.0.2.

> **Note:**
>
> Starting with Oracle Database 12c, Release 1 (12.1), non-CDB architecture is deprecated. It can be desupported in a future release.

The following table provides additional information about releases supported for downgrading. When using this table, also read about compatibility in "Checking for Incompatibilities When Downgrading Oracle Database."

Table 8-1 Supported Releases and Editions for Downgrading

Oracle Database Release or Edition	Downgradable (Yes/No)	Notes
18	Yes	No additional information at this time.
12.2.0.1 and 12.1.0.2	Yes	You cannot downgrade a database after you set the compatible initialization parameter to 12.1.0.2. You can downgrade a non-CDB from 19 to 18, or 12.2 to 12.1.0.2, or 11.2.0.4 (all supported upgrade releases). Install the latest Release Update, Release Revision, bundle patch or patch set update (BP or PSU) before you downgrade a CDB, or before you unplug and downgrade a PDB. Patches are available for download on My Oracle Support. Refer to My Oracle Support note 756671.1 to obtain the latest patch set. You cannot downgrade to releases earlier than the minimum compatibility setting for the new Oracle Database release.

Table 8-1 (Cont.) Supported Releases and Editions for Downgrading

Oracle Database Release or Edition	Downgradable (Yes/No)	Notes
Oracle Enterprise Manager	No	If you downgrade to an earlier supported release, then you must reconfigure Oracle Enterprise Manager controls. Before you start your upgrade, you must use the `emdwgrd` utility to save DB Control files and data, so that you can restore Oracle Enterprise Manager Database Control (DB Control) after a downgrade.
Oracle Database Express Edition	No	You cannot downgrade a database that is upgraded from Oracle Database Express Edition.

Recommendations to Review Before Downgrading

The following recommendations for earlier supported releases affect downgrading for Oracle Database:

- Multitenant architecture provides architecture features for a multitenant container database (CDB), and pluggable databases (PDBs). If you are upgrading to multitenant architecture, and you set the compatible initialization parameter to the highest level after upgrading to this release, then you cannot downgrade the database after an upgrade.
- This release contains a new object privilege, `READ`, in addition to `SELECT`. After you downgrade, note the following implications of this object privilege:
  - If you have the `SELECT` and `READ` object privileges, then the `READ` privilege is removed.
  - If you previously only had the `READ` object privilege, then the `READ` object privilege is transformed into the `SELECT` object privilege.

  Refer to Oracle Database Security Guide for more information about the READ and SELECT object privileges.
- If Oracle XML DB is not installed in the database that you upgrade, then during a downgrade, Oracle XML DB is uninstalled. For example, if you did not install Oracle XML DB with Oracle Database 11g Release 2 (11.2), then Oracle XML DB is installed with Oracle Database 12c. If you downgrade the database, then Oracle XML DB is uninstalled as part of the downgrade. Oracle XML DB is included by default with Oracle Database 12c release 1 (12.1), and later releases.
- During upgrade to Oracle Database 12c, the Database (DB) Control repository is removed. If you downgrade to an earlier release, then you must reconfigure the Database (DB) Control to use it after the downgrade.
- Downgrade is not supported for Oracle Enterprise Manager. If you downgrade to an earlier supported release, then you must reconfigure Oracle Enterprise Manager controls.

Related Topics

- https://support.oracle.com/rs?type=doc&id=756671.1
- *Oracle Database Security Guide*

Check for Incompatibilities When Downgrading Oracle Database

To see if the database has incompatibilities that can prevent you from downgrading, check the compatibility level of your database.

If you have updated the `COMPATIBLE` parameter to set the compatibility level of your Oracle Database release to the current release, then you are not able to downgrade to an earlier release. This issue occurs because new releases have changes to the Data Dictionary, and can have other feature changes that prevent downgrades.

To check the `COMPATIBLE` parameter setting for your database before you downgrade, enter the following command:

```
SQL> SELECT name, value, description FROM v$parameter WHERE name =
    'compatible';
```

> **Note:**
>
> For Oracle ASM disk groups, if you changed the `compatible.asm` parameter after the upgrade to the upgraded release value, then when you downgrade to the earlier release, you cannot mount your Oracle ASM disk groups. The value for `compatible.asm` sets the minimum Oracle ASM release that can mount a disk group.
>
> As part of your downgrade, you must create a new disk group to your downgraded release level, and restore data to that downgraded compatibility ASM disk group.

> **See Also:**
>
> *Oracle Automatic Storage Management Administrator's Guide* for information about Oracle ASM disk group compatibility

Perform a Full Backup Before Downgrading Oracle Database

Oracle strongly recommends that you perform a full backup of your new Oracle Database release before you downgrade to a supported earlier release.

> **See Also:**
>
> *Oracle Database Backup and Recovery User's Guide* for information about performing RMAN backups

Performing Required Predowngrade Steps for Oracle Database

Complete the required preparation steps described here before you downgrade Oracle Database to the earlier release from which you upgraded.

Before you start a downgrade, you must resolve incompatibilities between database releases. For example, determine if you must disable components in the database before you start the downgrade.

1. If you are downgrading a CDB or unplugging and downgrading a PDB in Oracle Database, then you must first apply the latest quarterly release update (Update), release update revision (Revision), bundle patch or patch set update (BP or PSU) available. Apply any required set of additional fixes on My Oracle Support Note 756671.1:

   https://support.oracle.com/rs?type=doc&id=756671.1

2. If you have enabled Oracle Database Vault on your database, then disable Oracle Database Vault before downgrading the database.

   Use DBA_DV_STATUS to find out if Oracle Database Vault is enabled:

   ```
   SQL> SELECT * FROM DBA_DV_STATUS;
   ```

   If the output is TRUE, then Oracle Database Vault is enabled, so you must disable it.

   On multitenant architecture Oracle Database systems, use `CDB_DV_STATUS` on `CDB$ROOT` to find out the Oracle Database Vault status on all PDBs plugged in to the CDB:

   ```
   SQL> SELECT * FROM CDB_DV_STATUS;
   ```

3. If you have enabled Unified Auditing, then you can choose to back up and purge the unified audit trail:

   a. Find if unified audit records exist.

      ```
      SQL> SELECT COUNT(*) FROM UNIFIED_AUDIT_TRAIL;
      ```

   b. Back up the existing audit data to a table. For example:

      ```
      SQL> CREATE TABLE UA_DATA AS (SELECT * FROM UNIFIED_AUDIT_TRAIL);
      ```

c. Clean up the audit trail.

```
EXEC DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL(audit_trail_type =>
DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED, use_last_arch_timestamp =>
FALSE);
```

4. Before downgrade, ensure that the target Oracle home for the downgraded database contains the version of the time zone file that your database is using.

   To find which time zone file version your database is currently using, query `V$TIMEZONE_FILE` using the following command:

   ```
   SQL> select * from  V$TIMEZONE_FILE;
   ```

   For example:

   If the query returns `timezlrg_20.dat` in the column `V$TIMEZONE_FILE.FILENAME`, then check if the file is present in the target Oracle Home:

   Linux and Unix

   ```
   $ORACLE_HOME/oracore/zoneinfo/timezlrg_20.dat
   ```

   Windows

   ```
   %ORACLE_HOME%\oracore\zoneinfo\timezlrg_20.dat
   ```

   If the required time zone file is missing from the target Oracle home, then do one of the following:

   - If you installed the current version of the time zone file as a patch, and you still know the patch number, then use the same patch number to download the corresponding time zone file for the target release from the My Oracle Support website.
   - Locate the correct patch by using the My Oracle Support website patch search function. Enter the following search criteria: "Product is 'Oracle Database'", "Release is '*target release*'", and "Description contains 'DST'".
   - If you cannot locate the patch on the My Oracle Support website, then log a service request with Oracle Support.

   After you find and download the required patch, install it in the target Oracle home.

5. If you created objects based on fixed objects, then drop these objects to avoid possible `ORA-00600` errors. You can recreate these objects after the downgrade.

6. If you have Oracle Enterprise Manager configured in your database, then drop the Enterprise Manager user:

   ```
   DROP USER sysman CASCADE;
   ```

> **Note:**
>
> After you drop the Enterprise Manager user, you can find that `MGMT*` synonyms are invalid. You must reconfigure Oracle Enterprise Manager to use any Oracle Enterprise Manager controls in the downgraded database.

Related Topics

- *Oracle Database Globalization Support Guide*

Using Scripts to Downgrade a CDB or Non-CDB Oracle Database

To automate downgrades, starting in Oracle Database 19c, Oracle provides the `dbdowngrade` utility script. When necessary, you can also continue to run `catdwgrd.sql` manually, as in previous releases.

- Using Dbdowngrade to Downgrade CDB and Non-CDB Oracle Databases
  To downgrade Oracle Database to an earlier supported major release, or to an earlier release update, Oracle recommends that you run the downgrade script `dbdowngrade`.
- Downgrading a CDB or Non-CDB Oracle Database Manually with catdwgrd.sql
  You can run the manual `catdwgrd.sql` script for downgrades of Oracle Database to an earlier supported major release, or an earlier release update, where you either prefer to run a manual script, or you want to avoid excessive thread issues.

Using Dbdowngrade to Downgrade CDB and Non-CDB Oracle Databases

To downgrade Oracle Database to an earlier supported major release, or to an earlier release update, Oracle recommends that you run the downgrade script `dbdowngrade`.

Starting with Oracle Database 19c, Oracle provides the Downgrade Utility script `dbdowngrade`. When you use the dbdowngrade utility, it sets appropriate values for the downgrade, and simplifies how you start a downgrade. Specifically, it ensures that the underlying calls to `catcon.pl` use recommended values, so that potential errors due to excessive threads being spawned are reduced. This feature is especially of value for downgrades of multitenant architecture (CDB) databases. When the potential of excessive threads exists, you can continue the downgrade by running the `catdwgrd.sql` script manually, as in previous releases.

The `dbdowngrade` shell command is located in the file path `$ORACLE_HOME/bin` on Linux and Unix, and `%ORACLE_HOME%\bin` on Microsoft Windows based systems. If you are downgrading a CDB, then you can provide the inclusion list as argument to the script.

When you downgrade multitenant architecture databases (CDBs), the `dbdowngrade` script has two behaviors, depending on whether you use an inclusion list.

- **Without an inclusion list**. The downgrade runs on all the containers that are open in the CDB (PDB and CDB).

Run the downgrade without an inclusion list when you want to downgrade the entire CDB. In this scenario, all open containers are downgraded. You must open all the PDBs in the CDB manually before you start the `dbdowngrade` script.

- **With an inclusion list**. The downgrade runs only on the PDBs within the inclusion list, and CDB$ROOT is not downgraded during the downgrade operation.

  Run the downgrade with an inclusion list when you want to downgrade only the set of PDBs listed in the inclusion list. In this scenario, where you want to use unplug and plug upgrades, only the set of PDBs that you list in the inclusion list are downgraded. The CDB and the PDBs that are not on the inclusion list remain upgraded to the later release.

Prerequisites:

- If you are downgrading from Oracle Database 19c to Oracle Database 18c, Oracle Database 12.2, or Oracle Database 12.1, then you can downgrade all databases in a multitenant container database (CDB), or one pluggable database (PDB) within a CDB. Oracle Database releases earlier than Oracle Database 12c did not use multitenant architecture.
- If you are downgrading without an inclusion list, then you must open all PDB containers before you run the `dbdowngrade` script.

1. Log in to the system as the owner of the Oracle Database Oracle home directory.
2. Set the `ORACLE_HOME` environment variable to the Oracle home of the upgraded Oracle Database release.
3. Set the `ORACLE_SID` environment variable to the system identifier (SID) of the Oracle Database that you want to downgrade.
4. At a system prompt, change to the directory *`ORACLE_HOME`*`/rdbms/admin`, where *`ORACLE_HOME`* is the Oracle home on your system.

> **Note:**
>
> If you are downgrading a cluster database, then shut down the database completely, and change the value for the initialization parameter `CLUSTER_DATABASE` to `FALSE`. After the downgrade, set this parameter back to `TRUE`.

5. Using SQL*Plus, connect to the database instance that you want to downgrade as a user with `SYSDBA` privileges:

```
sqlplus sys as sysdba
Enter password: password
```

6. Start the instance in downgrade mode by issuing the following SQL*Plus command for your Oracle Database instance type. In case of errors during startup, you can be required to use the `PFILE` option to specify the location of your initialization parameter file.
   - Non-CDB instances:

```
SQL> startup downgrade pfile=pfile_name
```

- CDB instances:

```
SQL> startup downgrade pfile=pfile_name
SQL> alter pluggable database all open downgrade;
```

Specify the location of your initialization parameter file `PFILE`.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for information about specifying initialization parameters at startup and the initialization parameter file

7. Start the `dbdowngrade` script, either with default values, or with an inclusion list.

   If you use an inclusion list, then CDB$ROOT must not be on your inclusion list.

   For example:

   `Running with default values`
   Linux and Unix

```
$cd $ORACLE_HOME/bin
$./dbdowngrade
```

   Microsoft Windows

```
$cd %ORACLE_HOME%\bin
$dbdowngrade.cmd
```

   `Running with inclusion list for CDB`
   Linux and Unix

```
$cd $ORACLE_HOME/bin
$./dbdowngrade –c 'PDB1 PDB2 PDBN'
```

   Microsoft Windows

```
$cd %ORACLE_HOME%\bin
$dbdowngrade.cmd –c "PDB1 PDB2 PDBN"
```

As a result of running the `dbdowngrade` script, the utility runs `catdwgrd` and `catcon.pl`. These scripts perform the downgrade, using the recommended values for the release to which you are downgrading. The log files produced by the downgrade scripts are placed under the first directory found of one of these three options, in order:

- The Oracle base home identified by the `orabasehome` command
- The Oracle base home identified by the `orabase` command
- The Oracle home identified by the `oracle_home` command

For example:

```
$ $ORACLE_HOME/bin/orabasehome
/u01/app/oracle/product/19.0.0/dbhome_1
```

In this example, the $ORACLE_BASE directory is `/u01/app/oracle/product/19.0.0/dbhome_1`, and the logs are located in `/u01/app/oracle/product/19.0.0/dbhome_1/cfgtoollogs/downgrade`. In the directory, the log files are prefixed with the string `catdwgrd`.

> **Note:**
>
> - Read-write Oracle homes: the commands `orabaseconfig` and `orabasehome` both return the environment setting for ORACLE_HOME.
> - Read-only Oracle homes: the command `orabaseconfig` returns the read-only path configuration for the Oracle base in the path `$ORACLE_BASE/homes`.

Downgrading a CDB or Non-CDB Oracle Database Manually with catdwgrd.sql

You can run the manual `catdwgrd.sql` script for downgrades of Oracle Database to an earlier supported major release, or an earlier release update, where you either prefer to run a manual script, or you want to avoid excessive thread issues.

If you are downgrading from Oracle Database 19c to Oracle Database 18c, Oracle Database 12.2, or Oracle Database 12.1, then you can downgrade all databases in a multitenant container database (CDB), or one pluggable database (PDB) within a CDB. Oracle Database releases earlier than Oracle Database 12c did not use multitenant architecture.

> **Note:**
>
> Starting with Oracle Database 12c release 1 (12.1), non-CDB architecture is deprecated. It can be desupported in a future release.

1. Log in to the system as the owner of the Oracle Database Oracle home directory.
2. Set the `ORACLE_HOME` environment variable to the Oracle home of the upgraded Oracle Database release.
3. Set the `ORACLE_SID` environment variable to the system identifier (SID) of the Oracle Database that you want to downgrade.
4. At a system prompt, change to the directory *`ORACLE_HOME`*`/rdbms/admin`, where *`ORACLE_HOME`* is the Oracle home on your system.

> **Note:**
>
> If you are downgrading a cluster database, then shut down the database completely, and change the value for the initialization parameter `CLUSTER_DATABASE` to `FALSE`. After the downgrade, set this parameter back to `TRUE`.

5. Using SQL*Plus, connect to the database instance that you want to downgrade as a user with `SYSDBA` privileges:

```
sqlplus sys as sysdba
Enter password: password
```

6. Start the instance in downgrade mode by issuing the following SQL*Plus command for your Oracle Database instance type. In case of errors during startup, you can be required to use the `PFILE` option to specify the location of your initialization parameter file.

   - Non-CDB instances:

```
SQL> startup downgrade pfile=pfile_name
```

   - CDB instances:

```
SQL> startup downgrade pfile=pfile_name
SQL> alter pluggable database all open downgrade;
```

   Specify the location of your initialization parameter file `PFILE`.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for information about specifying initialization parameters at startup and the initialization parameter file

7. (Recommended) If you are downgrading a non-CDB, then Oracle recommends that you set the system to spool results to a log file, so that you can track the changes and issues.

   If you are downgrading a CDB, then you do not need to perform this step. CDBs automatically spool output to the `catcon_logs`.

   On a non-CDB, enter the following command to spool results to a log file, where `downgrade.log` is the name of the log file:

```
set echo on termout on serveroutput on timing on
SQL> SPOOL downgrade.log
```

8. Use the following command to start the downgrade, depending on your configuration:

- Non-CDB:

```
SQL> @catdwgrd.sql
```

- CDB:

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -d
$ORACLE_HOME/rdbms/admin -e -b catdwgrd -l output directory -r
catdwgrd.sql
```

In the CDB example, `catdwgrd.sql` is run on containers using `catcon.pl`. To run commands with the `catcon.pl` utility, you first start Perl. The `-d` parameter tells `catcon.pl` where to find `catdwgrd`. The `-l` parameter specifies the output directory for log files (instead of writing to the `rdbms/admin` directory). Specifying the `-r` parameter causes `catdwgrd` to run first on the PDBs, and second on CDB_ROOT.

Run `catdwgrd` using the `-r` parameter when you downgrade a CDB. The `–r` parameter changes the default order that scripts are run, so that scripts run in all PDBs, and then in CDB_ROOT.

Note:

- Use the version of the `catdwgrd.sql` script included with your new Oracle Database release.
- Run `catdwgrd` using the `-r` parameter when downgrading a CDB.
- Run `catdwgrd.sql` in the new Oracle Database release environment.
- The `catdwgrd.sql` script downgrades all Oracle Database components in the database to the release from which you upgraded. The downgrade is either to the supported major release from which you upgraded, or to the patch release from which you upgraded.

If you are downgrading a multitenant environment database, and the `catdwgrd.sql` command encounters a failure, then review the error message. Check to see what issues are present in the `CDB$ROOT` or PDBs before proceeding. Check the section "Troubleshooting the Downgrade of Oracle Database." Fix the issues as stated in the errors. After you resolve the errors, rerun `catdgwrd.sq` with the `catcon.pl` utility, using the syntax `catcon.pl -c 'cdb,pdb' -r`.

Caution:

If the downgrade for a component fails, then an `ORA-39709` error is displayed. The SQL*Plus session terminates without downgrading the Oracle Database data dictionary. All components must be successfully downgraded before the Oracle Database data dictionary is downgraded. Identify and fix the problem before rerunning the `catdwgrd.sql` script.

9. For Non-CDB only, if you turned the spool on, then turn off the spooling of script results to the log file:

```
SQL> SPOOL OFF
```

Check the spool file, and verify that no errors occurred during the downgrade. You named the spool file in Step 8, and the suggested name was `downgrade.log`. Correct any problems that you find in this file. If necessary, rerun the downgrade script.

> **Note:**
>
> Save the results from the first time you ran the downgrade script. Before you rerun the downgrade script, rename the file `downgrade.log` to a different name, so that it is not overwritten when you rerun the script.

10. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

11. Exit SQL*Plus.
12. If your operating system is Linux or Unix, then change the following environment variables to point to the directories of the release to which you are downgrading:
    - Linux and UNIX systems

      Change the following environment variables to point to the directories of the release to which you are downgrading:

      – `ORACLE_HOME`
      – `PATH`

      Also check that your `oratab` file, and any client scripts that set the value of `ORACLE_HOME`, point to the downgraded Oracle home.

      > **See Also:**
      >
      > *Oracle Database Installation Guide* for your operating system for information about setting other important environment variables on your operating system

    - Microsoft Windows systems

      a. Stop all Oracle services, including the `OracleService`*`SID`* Oracle service of the Oracle Database 12c 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:**
>
> *Oracle Database Administrator's Reference for Microsoft Windows* for more information about stopping Oracle services on Windows

b. Delete the Oracle service at a command prompt by issuing the command `ORADIM`.

For example, if your *SID* is `ORCL`, then enter the following command:

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

Create the Oracle service of the database that you are downgrading at a command prompt using the command `ORADIM`:

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

The syntax for `ORADIM` includes the following variables:

Variable	Description
SID	Same system identifier (SID) name as the SID of the database being downgraded.
PASSWORD	Password for the database instance. This password is the password for the user connected with SYSDBA privileges. The `-INTPWD` option is not required. If you are prompted for a password, then use the password for the standard user account for this Windows platform.
USERS	Maximum number of users that can be granted SYSDBA and SYSOPER privileges.
ORACLE_HOME	Oracle home directory of the database to which you are downgrading. Ensure that you specify the full path name with the option `-PFILE`, including the drive letter where the Oracle home directory is mounted. See *Oracle Database Administrator's Guide* for information about specifying initialization parameters at startup, and for information about the initialization parameter file.

For example, if your *SID* is `ORCL`, your *PASSWORD* is `TWxy5791`, 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 TWxy5791 -MAXUSERS 10
     -STARTMODE AUTO -PFILE C:\ORANT\DATABASE\INITORCL.ORA
```

> **Note:**
>
> The `ORADIM` command prompts you for the password for the Oracle home user account. You can specify other options using `ORADIM`.
>
> You are not required to change any Windows Registry settings when downgrading a database. The `ORADIM` utility makes all necessary changes automatically.

> **See Also:**
>
> *Oracle Database Administrator's Reference for Microsoft Windows* for information about administering an Oracle Database instance using `ORADIM`

13. Restore the configuration files (for example, parameter files, password files, and so on) of the release to which you are downgrading.

    If the database is an Oracle RAC database, then run the following command to return the database to single instance mode:

    ```
    SET CLUSTER_DATABASE=FALSE
    ```

    > **Note:**
    >
    > If you are downgrading a cluster database, then perform this step on all nodes on which this cluster database has instances configured. Set the value for the initialization parameter `CLUSTER_DATABASE` to `FALSE`. After the downgrade, set this initialization parameter back to `TRUE`.

    > **See Also:**
    >
    > *Oracle Real Application Clusters Administration and Deployment Guide* for information about initialization parameter use in Oracle RAC

14. At a system prompt, change to the `admin` directory in the Oracle home directory of the earlier release to which you are downgrading. (*`ORACLE_HOME`*`/rdbms/admin`, where *`ORACLE_HOME`* is the path to the earlier release Oracle home.)

15. Start SQL*Plus, and connect to the database instance as a user with SYSDBA privileges.

    - For a non-CDB:

      ```
      SQL> CONNECT / AS SYSDBA
      SQL> STARTUP UPGRADE
      ```

- For a CDB:

```
connect / as sysdba
startup upgrade;
alter pluggable database all open upgrade;
```

16. (Optional) For a non-CDB, set the system to spool results to a log file to track changes and issues. This step is not needed for a CDB.

```
SQL> SPOOL reload.log
```

17. Run `catrelod.sql` on non-CDB databases, or use `catcon.pl` to run `catrelod.sql` on CDB databases.

- For a non-CDB:

```
SQL> $ORACLE_HOME/rdbms/admin/catrelod.sql
```

- For a CDB:

```
 $ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -
n 1 -e -b catrelod -d $ORACLE_HOME/rdbms/admin catrelod.sql
```

This command reloads the appropriate version for each of the database components in the downgraded database.

18. If you turned on spooling for a non-CDB, then turn off the spooling of script results to the log file:

```
SQL> SPOOL OFF
```

Check the spool file, and verify that the packages and procedures compiled successfully. Correct any problems that you find in this log file, and rerun the appropriate script, if necessary.

19. Shut down and restart the instance for normal operation:

```
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
```

You can be required to use the option`PFILE` to specify the location of your initialization parameter file.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for information about specifying initialization parameters at startup, and in the initialization parameter file

20. If you configured your database to use Oracle Label Security, then complete this step. If you did not configure your database to use Oracle Label Security, then proceed to the next step.

a. Copy the script `olstrig.sql` from the Oracle home under Oracle Database 12c to the Oracle home of the release number to which you are downgrading the database.

b. From the Oracle home of the downgrade release, run `olstrig.sql` to recreate DML triggers on tables with Oracle Label Security policies:

```
SQL> @olstrig.sql
```

21. (Optional) For a non-CDB, set the system to spool results to a log file to track changes and issues. This step is not needed for a CDB.

    Example:

    ```
    SQL> SPOOL utlrp.log
    ```

22. Run the `utlrp.sql` script to recompile any remaining stored PL/SQL and Java code. Use the procedure for your configuration:

    - non-CDB:

      ```
      SQL> $ORACLE_HOME/rdbms/admin/utlrp.sql
      ```

    - CDB:

      ```
      $ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -n 1 -e -b
      utlrp -d $ORACLE_HOME/rdbms/admin utlrp.sql
      ```

      The `utlrp.sql` script recompiles all existing PL/SQL modules previously in `INVALID` state, such as packages, procedures, types, and so on. The log file `utlrp0.log` is generated. That log file lists the recompilation results.

23. If you turn on spooling for a non-CDB when you run `utlrp.sql`, then turn off the spooling of script results to the log file:

    ```
    SQL> SPOOL OFF
    ```

    Check the spool file, and verify that the packages and procedures compiled successfully. Correct any problems that you find in this log file. If necessary, rerun the appropriate script.

24. Exit SQL*Plus.

25. If you are downgrading a cluster database, then you must run the following command to downgrade the Oracle Clusterware database configuration:

    ```
    $ srvctl downgrade database -d db-unique-name -o oraclehome -t
    to_version
    ```

    Replace the variables in this syntax example with the values for your system:

    - *db-unique-name* is the database name (not the instance name).
    - *oraclehome* is the location of the old Oracle home for the downgraded database.
    - *to_version* is the database release to which the database is downgraded. (For example: 12.2.0.1.)

> **Note:**
>
> Run this command from the new release Oracle home, not from the Oracle home to which the database is being downgraded.

At the completion of this procedure, your database is downgraded.

Related Topics

- Troubleshooting the Downgrade of Oracle Database
- *Oracle Database Administrator's Guide*

About Downgrades and Invalid Objects with Component Status OPTION OFF

Downgrades from later release Oracle Database Standard Edition to Oracle Database 12c Standard Edition with CDB and PDBS contain later release invalid objects after the downgrade.

With downgrades from later release Oracle Database releases to Oracle Database Release 12c (12.2 or 12.1), it is expected behavior to find later release invalid objects in the downgraded database, where the Oracle Database Release 12c database had these objects set to FALSE. This problem does not exist with downgrades to Oracle Database Release 18c (18.1.0) and later Oracle Database releases.

For example:

During a downgrade from Oracle Database 19c Standard Edition to Oracle Database 12c Release 2 (12.2.0.1) Standard Edition, you have OLAP and Spatial components upgraded to the 19c release versions, though they are set to FALSE. After the downgrade, the OLAP and Spatial components are still not available, but they are not reloaded into the 12.2.0.1 release. You then see this result in the DBA registry, where you have later release objects listed in your 12.2 registry:

```
OLAP Analytic Workspace            OPTION OFF   19.1.0
Oracle OLAP API                    OPTION OFF   19.1.0
```

This result is expected behavior in downgrades from Oracle Database 18c and later to Oracle Database 12c. The reason that this behavior is expected is because the option setting for these options in Standard Edition databases is FALSE. During the upgrade, all options are upgraded, even though they are set to `FALSE`. During the run of `catrelod.sql`, the reload scripts do not run to replace these objects with the earlier release versions. As a result, downgrading back to Standard Edition leaves many OLAP and Spatial objects at their later release versions.

Downgrading a Single Pluggable Oracle Database (PDB)

If you are downgrading Oracle Database, then you can downgrade one PDB without downgrading the whole CDB.

In Oracle Database releases later than Oracle Database 12c release 2 (12.2), you can downgrade individual PDBs. For example, you can unplug a PDB from an upgraded

CDB, downgrade the PDB, and then plug it in to an earlier release CDB, or you can convert the PDB database to a standalone database.

Downgrade the PDB

In this procedure example, you downgrade the PDB to release 12.1.0.2:

1. Start up the PDB in DOWNGRADE mode. The CDB can be in normal mode when you do this.

   ```
   SQL> alter pluggable database CDB1_PDB1 open downgrade;
   ```

2. Downgrade the PDB, either by using the `dbdowngrade` utility, or by running `catdwgrd` manually, using `catcon.pl`.

   In each of these options, the PDB that you are downgrading is `PDB1`.

   - Downgrading with `dbdowngrade` utility.

     Downgrade the PDB using the dbdowngrade script as follows:

     ```
     cd $ORACLE_HOME/bin
     ./dbdowngrade -c 'PDB1'
     ```

   - Manually downgrading with `catdwgrd`, using `catcon.p.`

     Run `catdwgrd` as follows:

     ```
     $ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -d
      $ORACLE_HOME/rdbms/admin -n 1 -l <output directory> -e -b catdwgrd -c
     'PDB1'
      catdwgrd.sql
     ```

     In the example, `catdwgrd` is run with `catcon.pl`. The `-d` parameter tells `catcon.pl` where to find `catdwgrd`. The `-l` parameter specifies the output directory for log files, instead of writing to the `rdbms/admin` directory). You must use the `-r` parameter to run the two scripts together at the same time.

     The log files for the downgrade are placed in the Oracle base home (the Oracle base identified by the commands `orabasehome`, or `orabase`, or the Oracle home identified by the command `oracle_home`, in that order.

3. Close the PDB.

Unplug the PDB from the CDB

In this step you unplug the downgraded PDB from the release 12.2.0.1 CDB:

1. Connect to the upgraded CDB.
2. Close the PDB that you want to unplug.

   ```
   SQL> alter pluggable database PDB1 close;
   ```

3. Unplug the downgraded 12.1.0.2 PDB, replacing the variable *`path`* with the path on your system:

   ```
   SQL> alter pluggable database PDB1 unplug into 'path/pdb1.xml';
   ```

   You receive the following response when the unplug is completed:

   ```
   Pluggable database altered
   ```

Plug in the Downgraded 12.1.0.2 PDB

In this step you plug the downgraded 12.1.0.2 PDB into the 12.1.0.2 CDB. To do this, you must create the PDB in this CDB. The following example shows how to create a pluggable database called PDB1:

1. Connect to the 12.1.0.2 CDB.
2. Plug in the 12.1.0.2 PDB.

   ```
   SQL> create pluggable database PDB1 using 'path/pdb1.xml';
   ```

   This command returns `Pluggable database created`.
3. Open the PDB in upgrade mode:

   ```
   SQL> alter pluggable database PDB1 open upgrade;
   ```
4. Connect to the PDB:

   ```
   SQL> alter session set container=PDB1;
   ```
5. Run `catrelod` in the PDB:

   ```
   SQL> @$ORACLE_HOME/rdbms/admin/catrelod.sql
   ```

   The `catrelod.sql` script reloads the appropriate version for each of the database components in the downgraded database.
6. Run `utlrp` in the PDB:

   ```
   SQL> @$ORACLE_HOME/rdbms/admin/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.

Downgrading PDBs That Contain Oracle Application Express

Use this procedure to avoid INVALID OBJECTS OWNED BY APEX_050000 errors when you downgrade PDBs that contain Oracle Application Express.

After you downgrade the PDB to an earlier release, enter a SQL statement similar to the following to drop the Oracle Application Express user:

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -b
drop_apex5
-c 'PDB1' -- --x'drop user apex_050000 cascade'
```

In this example, the PDB name is `'PDB1'`.

Post-Downgrade Tasks for Oracle Database Downgrades

After you downgrade your Oracle Database release, you can be required to complete additional tasks, due to changes that affect compatibility, components, and supported protocols.

- Oracle XML DB Authentication Recommendations for an Oracle Database Downgrade
  Upgrades from releases earlier than 12.1 result in digest authentication being disabled.
- Reapply Release Update and Other Patches After Downgrade
  After the downgrade is run, and `catrelod.sql` completes successfully, if you installed new patches in your original Oracle home after the upgrade, but before the downgrade, then ensure that you apply any patches that you installed.
- Re-enabling Oracle Database Vault after Downgrading Oracle Database
  You must do this if you are instructed during the downgrade to disable Oracle Database Vault.
- Restoring the Configuration for Oracle Clusterware
  To restore the configuration, you must restore the release from which you upgraded.
- Restoring Oracle Enterprise Manager after Downgrading Oracle Database
  The restore task described in this section is required only if you are performing a downgrade, and Oracle Enterprise Manager is configured on the host.
- Restoring Oracle Application Express to the Earlier Release
  After a downgrade, if you upgraded Oracle Application Express at the same time as you upgraded Oracle Database, then you must complete steps to revert to the earlier Oracle Application Express release.
- Gathering Dictionary Statistics After Downgrading
  To help to assure good performance after you downgrade, use this procedure to gather dictionary statistics.
- Regathering Fixed Object Statistics After Downgrading
  After the downgrade, run representative workloads on Oracle Database, and regather fixed object statistics.
- Regathering Stale CBO Statistics After Downgrade
  Oracle recommends that you regather Oracle Cost-Based Optimizer (CBO) statistics after completing an Oracle Database downgrade.

Oracle XML DB Authentication Recommendations for an Oracle Database Downgrade

Upgrades from releases earlier than 12.1 result in digest authentication being disabled.

If you downgrade to a release that is earlier than Oracle Database 12c, in which digest authentication is not supported, digest authentication is disabled and made unavailable as an authentication choice. This affects HTTP authentication for Oracle XML DB Repository. If you did not take advantage of digest authentication and instead used the default configuration, then no further actions are necessary.

Reapply Release Update and Other Patches After Downgrade

After the downgrade is run, and `catrelod.sql` completes successfully, if you installed new patches in your original Oracle home after the upgrade, but before the downgrade, then ensure that you apply any patches that you installed.

If you installed new patches, then run the `datapatch` tool to apply those patches to the downgraded database. If you did not change the binaries and files in your Oracle Home after the upgrade, then there is no need to run `datapatch` after running `catrelod.sql`. However, if you are in any doubt about whether new patches are installed, then run `datapatch`. There is no safety concern that prevents you from running `datapatch` as many times as you require to be certain that patches are applied to the database.

Re-enabling Oracle Database Vault after Downgrading Oracle Database

You must do this if you are instructed during the downgrade to disable Oracle Database Vault.

If you use Oracle Database Vault, then you may have been instructed to disable it before downgrading your database. To use Oracle Database Vault after downgrading, you must re-enable it.

> **See Also:**
>
> *Oracle Database Vault Administrator's Guide* for the procedure to re-enable Oracle Database Vault

Restoring the Configuration for Oracle Clusterware

To restore the configuration, you must restore the release from which you upgraded.

You can restore the Oracle Clusterware configuration to the state it was in before the Oracle Clusterware upgrade. Any configuration changes that you have performed during or after the new Oracle Database upgrade process are removed, and cannot be recovered.

Restoring Oracle Enterprise Manager after Downgrading Oracle Database

The restore task described in this section is required only if you are performing a downgrade, and Oracle Enterprise Manager is configured on the host.

To restore Oracle Enterprise Manager, you first run Oracle Enterprise Manager configuration assistant (EMCA), and then you run the `emdwgrd` utility.

- Requirements for Restoring Oracle Enterprise Manager After Downgrading
  You must complete these requirements before you upgrade to be able to restore Oracle Enterprise Manager after a downgrade to a release earlier than 12.1
- Running EMCA to Restore Oracle Enterprise Manager After Downgrading
  Review these topics and select your restoration scenario to restore Oracle Enterprise Manager after a downgrade.

- Running the emdwgrd utility to restore Enterprise Manager Database Control
  You can restore the Oracle Enterprise Manager Database Control and data by using the `emdwgrd` utility after you run `emca -restore`.

Requirements for Restoring Oracle Enterprise Manager After Downgrading

You must complete these requirements before you upgrade to be able to restore Oracle Enterprise Manager after a downgrade to a release earlier than 12.1

The following must be true to use `emca -restore` to restore Oracle Enterprise Manager to its previous state:

- Before the upgrade, you saved a backup of your Oracle Enterprise Manager configuration files and data
- You run the `emca` binary located in the new Oracle Database release home for this procedure

On Oracle Clusterware systems, to restore Oracle Enterprise Manager on an Oracle RAC database, you must have the database registered using `srvctl` before you run `emca -restore`. You must run `emca -restore` from the *`ORACLE_HOME`*`/bin` directory of the earlier Oracle Database release to which the database is being downgraded.

Run the `emca -restore` command with the appropriate options to restore Oracle Enterprise Manager Database Control or Grid Control to the old Oracle home.

Specify different `emca` options, depending on whether the database you want to downgrade is a single-instance database, an Oracle RAC database, or an Oracle ASM database.

Related Topics

- *Oracle Clusterware Administration and Deployment Guide*

Running EMCA to Restore Oracle Enterprise Manager After Downgrading

Review these topics and select your restoration scenario to restore Oracle Enterprise Manager after a downgrade.

- Running emca on a Single-Instance Oracle Database Without Oracle ASM
  Use Enterprise Manager Configuration Assistant (`emca`) to manage your database.
- Running EMCA on an Oracle RAC Database Without Oracle ASM
  Use Enterprise Manager Configuration Assistant (`emca`) to manage your database:
- Running EMCA on a Single-Instance Oracle ASM Instance
  Use Enterprise Manager Configuration Assistant (`emca`) to manage your database and storage.
- Running emca on an Oracle ASM on Oracle RAC Instance
  Use Enterprise Manager Configuration Assistant (`emca`) to manage your database and storage.
- Running emca on a Single-Instance Oracle Database With Oracle ASM
  Use Enterprise Manager Configuration Assistant (`emca`) to manage your database and storage.

- Running emca on an Oracle RAC Database and Oracle ASM Instance
  Use Enterprise Manager Configuration Assistant (emca) to manage your database and storage.

Running emca on a Single-Instance Oracle Database Without Oracle ASM

Use Enterprise Manager Configuration Assistant (emca) to manage your database.

Use this command to run Enterprise Manager Configuration Assistant.

```
ORACLE_HOME/bin/emca -restore db
```

You are prompted to enter the following information:

- Oracle home for the database that you want to restore
- Database SID
- Listener port number

Running EMCA on an Oracle RAC Database Without Oracle ASM

Use Enterprise Manager Configuration Assistant (emca) to manage your database:

Use this procedure to run Enterprise Manager Configuration Assistant:

```
ORACLE_HOME/bin/emca -restore db -cluster
```

You are prompted to enter the following information:

- Oracle home for the database that you want to restore
- Database unique name
- Listener port number

Running EMCA on a Single-Instance Oracle ASM Instance

Use Enterprise Manager Configuration Assistant (emca) to manage your database and storage.

Use this command to run Enterprise Manager Configuration Assistant.

```
ORACLE_HOME/bin/emca -restore asm
```

You are prompted to enter the following information:

- Oracle home for the database that you want to restore
- Oracle ASM port
- Oracle ASM SID

Running emca on an Oracle ASM on Oracle RAC Instance

Use Enterprise Manager Configuration Assistant (emca) to manage your database and storage.

Use this command to run Enterprise Manager Configuration Assistant.

```
ORACLE_HOME/bin/emca -restore asm -cluster
```

You are prompted to enter the following information:

- Oracle home for the database that you want to restore
- Oracle ASM port

Running emca on a Single-Instance Oracle Database With Oracle ASM

Use Enterprise Manager Configuration Assistant (`emca`) to manage your database and storage.

Use this command to run Enterprise Manager Configuration Assistant.

```
ORACLE_HOME/bin/emca -restore db_asm
```

You are prompted to enter the following information:

- Oracle home for the Oracle Database that you want to restore
- Database SID
- Listener port number
- Oracle ASM port
- Oracle ASM home
- Oracle ASM SID [+ASM]

Running emca on an Oracle RAC Database and Oracle ASM Instance

Use Enterprise Manager Configuration Assistant (`emca`) to manage your database and storage.

Use this command to run Enterprise Manager Configuration Assistant:

```
ORACLE_HOME/bin/emca -restore db_asm -cluster
```

You are prompted to enter the following information:

- Oracle home for the database that you want to restore
- Database unique name
- Listener port number
- Oracle ASM port
- Oracle ASM Oracle home
- Oracle ASM SID [+ASM]

The output of `emca` varies according to the options that you specify and the values that you enter at the prompts. In Oracle RAC environments, you must repeat this step on all Oracle RAC cluster member nodes.

You must now run the `emdwgrd` utility to restore Oracle Enterprise Manager Database Control and data.

Running the emdwgrd utility to restore Enterprise Manager Database Control

You can restore the Oracle Enterprise Manager Database Control and data by using the `emdwgrd` utility after you run `emca -restore`.

To use `emdwgrd`, you must do the following:

- Set *ORACLE_HOME* and other environment variables to point to the Oracle home from which the upgrade originally took place.
- Run the `emdwgrd` utility from the new release Oracle Database Oracle home.

The following procedure is for Linux and Unix. To run it on Windows, substitute `emdwgrd.bat` for `emdwgrd`.

1. Set *ORACLE_HOME* to the Oracle home from which the database upgrade originally took place.
2. Set *ORACLE_SID* to the SID of the database that was upgraded and then downgraded.
3. Set *PATH*, *LD_LIBRARY_PATH* and *SHLIB_PATH* to point to the Oracle home from which the database upgrade originally took place.
4. Go to the new Oracle Database release Oracle home:

   ```
   cd $ORACLE_HOME/bin
   ```

5. Run `emdwgrd` using one of the following procedures:
   a. For a single-instance database, run the following command, where *`SID`* is the SID of the database that was upgraded and then downgraded and *`save_directory`* is the path to the storage location you chose when saving your database control files and data:

      ```
      emdwgrd -restore -sid SID -path save_directory -tempTablespace
      TEMP
      ```

   b. For an Oracle RAC database, remote copy is required across the cluster nodes. Define an environment variable to indicate which remote copy is configured. For example:

      ```
      setenv EM_REMCP /usr/bin/scp
      ```

      Then, run `emdwgrd –restore` with the following options:

      ```
      emdwgrd -restore -tempTablespace TEMP -cluster -sid SID_OldHome
      -path save_directory
      ```

      If the Oracle home is on a shared device, then add `-shared` to the `emdwgrd` command options.
6. Enter the SYS and SYSMAN passwords when prompted by `emdwgrd`.
7. When `emdwgrd` completes, Oracle Enterprise Manager Database Control is downgraded to the old Oracle home.

Restoring Oracle Application Express to the Earlier Release

After a downgrade, if you upgraded Oracle Application Express at the same time as you upgraded Oracle Database, then you must complete steps to revert to the earlier Oracle Application Express release.

To complete the downgrade of Oracle Application Express after a database downgrade, complete all the steps listed in *Oracle Application Express Installation Guide* to revert your Oracle Application Express release to the earlier release. The steps to revert are different, depending on whether your architecture is a Non-CDB or a multitenant architecture (CDB) Oracle Database.

> **Note:**
>
> You only need to complete these steps if you upgraded Oracle Application Express at the same time that you upgraded the database.

Related Topics

- *Oracle Application Express Installation Guide*

Gathering Dictionary Statistics After Downgrading

To help to assure good performance after you downgrade, use this procedure to gather dictionary statistics.

Oracle recommends that you gather dictionary statistics after downgrading the database, so that the statistics are collected for the downgraded release Data Dictionary tables.

> **Note:**
>
> After a downgrade process, be aware that the the data dictionary can have changes that persist in the downgraded dictionary. These changes are insignificant. The downgraded data dictionary is functionally equivalent to an earlier release data dictionary.

- Non-CDB Oracle Database: Oracle recommends that you use the `DBMS_STATS.GATHER_DICTIONARY_STATS` procedure to gather these statistics. For example, enter the following SQL statement:

  ```
  SQL> EXEC DBMS_STATS.GATHER_DICTIONARY_STATS;
  ```

- CDB (multitenant architecture) Oracle Database: Oracle recommends that you use `catcon` to gather Data Dictionary statistics across the entire multitenant architecture.

  To gather dictionary statistics for all PDBs in a container database, use the following syntax:

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -l /tmp
-b gatherstats -- --x"exec dbms_stats.gather_dictionary_stats"
```

To gather dictionary statistics on a particular PDB, use syntax similar to the following:

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -
l /tmp -c
'SALES1' -b gatherstats -- --x"exec
dbms_stats.gather_dictionary_stats"
```

In the preceding example the `-c SALES1` option specifies a PDB inclusion list for the command that you run, specifying the database named `SALES1`. The option `-b gatherstats`specifies the base name for the logs. The option `--x` specifies the SQL command that you want to execute. The SQL command itself is inside the quotation marks.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

Regathering Fixed Object Statistics After Downgrading

After the downgrade, run representative workloads on Oracle Database, and regather fixed object statistics.

Fixed objects are the X$ tables and their indexes. V$ performance views are defined through X$ tables. After you downgrade, regather fixed object statistics to ensure that the optimizer for the restored database can generate good execution plans. These execution plans can improve database performance. Failing to obtain representative statistics can lead to suboptimal execution plans, which can cause performance problems

Gather fixed objects statistics by using the `DBMS_STATS.GATHER_FIXED_OBJECTS_STATS` PL/SQL procedure. `DBMS_STATS.GATHER_FIXED_OBJECTS_STATS` also displays recommendations for removing all hidden or underscore parameters and events from `init.ora` and `SPFILE`.

To gather statistics for fixed objects, run the following PL/SQL procedure:

```
SQL> execute dbms_stats.gather_fixed_objects_stats;
```

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for more information about using the `GATHER_FIXED_OBJECTS_STATS` procedure

Regathering Stale CBO Statistics After Downgrade

Oracle recommends that you regather Oracle Cost-Based Optimizer (CBO) statistics after completing an Oracle Database downgrade.

When you upgrade Oracle Database and gather new CBO statistics, the upgraded database has new database statistics. The upgraded database also can include new histogram types. For this reason, when you downgrade the database, the statistics that you collected for the new release can be different from the previous release. This issue is applicable both to data dictionary tables, and to regular user tables.

Regather stale statistics either by using GATHER_DATABASE_STATS, or by using gather commands that you typically use to update stale statistics in the dictionary and application schemas.

For example:

- Non-CDB Oracle Database: To regather statistics, Oracle recommends that you use the `GATHER_DATABASE_STATS` procedure, with the option `'GATHER STALE'`. For example:

  ```
  SQL> execute dbms_stats.gather_database_stats(options=>'GATHER STALE');
  ```

- CDB (multitenant architecture) Oracle Database: to regather Data Dictionary statistics across the entire multitenant architecture, Oracle recommends that you use `catcon`.

  To regather stale dictionary statistics for all PDBs in a container database, use the following syntax:

  ```
  $ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl
  -l /tmp -b gatherstats -- --x"exec
  dbms_stats.gather_database_stats(options=>'GATHER STALE')"
  ```

  To gather dictionary statistics on a particular PDB, use syntax similar to the following:

  ```
  $ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -
  l /tmp -c
  'SALES1' -b gatherstats -- --x"exec
  dbms_stats.gather_database_stats(options=>'GATHER STALE')"
  ```

  In the preceding example, the `-c SALES1` option specifies a PDB inclusion list for the command that you run, specifying the database named `SALES1`. The option `-b gatherstats`specifies the base name for the logs. The option `--x` specifies the SQL command that you want to execute. The SQL command itself is inside the quotation marks.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

Troubleshooting the Downgrade of Oracle Database

Use this troubleshooting information to address issues that may occur when downgrading Oracle Database.

This section contains known errors that may occur during downgrades, and workarounds to address those errors.

- Errors Downgrading Oracle Database Components with catdwgrd.sql Script
  Use this section to troubleshoot errors when you run the `catdwgrd.sql` script during a downgrade, such as `ORA-20001`: Downgrade cannot proceed.
- Oracle Multimedia Downgrade and imrelod.sql Script Error
  Review if you encounter ORA-20000: Oracle ORDIM component in registry is status: DOWNGRADED.
- Oracle Database Vault and dvrelod.sql Script Error
  Review if you encounter ORA-31011: XML parsing failed.
- Downgrading Oracle Grid Infrastructure (Oracle Restart) After Successful or Failed Upgrade
  To downgrade Oracle Restart, you must deconfigure and then reinstall Oracle Grid Infrastructure. You can then add back the databases and services.
- Oracle ACFS and Oracle Grid Infrastructure Downgrades to 11g Release 2 (11.2)
  You must run `acfsroot install` before you attempt to start the software stack.
- Database Links Passwords After Downgrading Oracle Database 11g Release 1 (11.1)
  Reset the passwords for any database links that were created in releases 11.2 or 12.1.

Errors Downgrading Oracle Database Components with catdwgrd.sql Script

Use this section to troubleshoot errors when you run the `catdwgrd.sql` script during a downgrade, such as `ORA-20001`: Downgrade cannot proceed.

The `catdwgrd.sql` script downgrades all Oracle Database components in the database to the major release from which you originally upgraded. This script must run before the Data Dictionary can be downgraded. If you encounter any problems when you run the script, then correct the causes of the problems, and rerun the script.

Errors you can see include `ORA-39709`: incomplete component downgrade; string downgrade aborted, and `ORA-06512`. When these errors occur, downgrades cannot proceed.

- **Cause**: One or more components that must be downgraded before proceeding with the Data Dictionary downgrade did not downgrade.
- **Action**: Review the log files to determine what errors occurred before the `catdwgrd.sql` script halted, and the downgrade was stopped.

Review these examples to understand how to correct this issue.

Errors typically describe what you must do to fix the issue that is preventing the downgrade to complete. Follow the instructions in the error message. After you have fixed the cause of the error, rerun the `catdwgrd.sql` script.

For example, If the CDB downgrade fails during the downgrade of CDB$ROOT due to a check, then follow the instructions in the error message to fix the condition error. After you fix the error, rerun `catdwgrd.sql` with `catcon.pl`. Use the `-c` option to run

the command with the inclusion list `'CDB$ROOT PDB1'`. Use the `-r` option to run the command first on the PDB, and then on CDB$ROOT. For example:

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -d
$ORACLE_HOME/rdbms/admin -e -b catdwgrd -l /scratch/rac/downgradeLogs
-c 'CDB$ROOT, PDB1, PDB2' -r catdwgrd.sql
```

Example 8-1 ORA-20001 Error Due To ORA-06512

Your downgrade stops. When you review the log files, you find that `catdwgrd.sql` terminates on this error:

```
DECLARE * ERROR at line 1: ORA-20001: Downgrade cannot proceed -
Unified Audit Trail data exists. Please clean up the data first
using DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL. ORA-06512: at line 65
ORA-06512: at line 42
```

You must purge the unified audit trial on `CDB$ROOT` and on all PDBs.

For example:

1. Look for the presence of unified audit trails:

   ```
   SQL> SELECT COUNT(*) FROM UNIFIED_AUDIT_TRAIL;
   COUNT(*)
   ----------
         4538
   ```

2. Purge the audit trail. on the CDB.

   For example, where the audit trail type is `DBMS_AUDIT.MGMT.AUDIT`:

   ```
   EXEC DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL DBMS_AUDIT_MGMT.AUDIT
   ```

3. Run `catdwngrd.sql` on `CDB$ROOT`. If PDBs still have unified audit data, then the script fails with ORA20001:

   ```
    62        execute immediate
    63        'select count(*) from audsys.'||'"'||tab_name||'"' into
   no_rows;
    64
    65        -- If audit trail has some data, raise the application
   error
    66        IF no_rows > 0 THEN
    67          RAISE_APPLICATION_ERROR(-20001, ErrMsg);
    68        END IF;
    69      END IF;
    70    END IF;
    71  EXCEPTION
    72    WHEN NO_DATA_FOUND THEN
    73     NULL;
    74    WHEN OTHERS THEN
    75     RAISE;
    76  END;
    77  /
   ```

```
DECLARE
*
ERROR at line 1:
ORA-20001: Downgrade cannot proceed - Unified Audit Trail data
exists.Please
clean up the data first using DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL.
ORA-06512: at line 75
```

4. Connect to individual PDBs, and find if they have unified audit trails. Clear the unified audit trail for all PDBs. For example, The PDB named PDB1 has unified audit trails:

```
ALTER SESSION SET container = PDB1;

SQL> SELECT COUNT(*) FROM UNIFIED_AUDIT_TRAIL;
 COUNT(*)
----------
      1330
```

5. Identify the unified audit trails:

```
SQL> CREATE TABLE UA_DATA AS (SELECT * FROM UNIFIED_AUDIT_TRAIL);
```

6. Purge the audit trails.

   In this example, the audit trail type is `DBMS_AUDIT_MGMT.AAUDIT_TRAIL_UNIFIED`, the `USE_LAST_ARCH_TIMESTAMP` value is set to `FALSE`, so that all audit records are deleted, without considering last archive timestamp, and the `CONTAINER` value is set to `DBMS_AUDIT_MGMT.CONTAINER_ALL`, so that audit records on all PDBs are purged.

```
BEGIN
  DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL(
   AUDIT_TRAIL_TYPE           =>
DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
   USE_LAST_ARCH_TIMESTAMP    =>  FALSE,
   CONTAINER                  =>  DBMS_AUDIT_MGMT.CONTAINER_ALL
END;
/
```

7. Rerun `catdwngrd.sql` at the PDB and CDB level. For example:

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -c
'CDB$ROOT,PDB1' -d $ORACLE_HOME/rdbms/admin -e -b catdwgrd -l /u01/
oracle/product/19.0.0/downgrade_logs -r catdwgrd.sql
```

8. Repeat the process of finding and purging audit trails and run `catdwgrd.sql` until the script completes successfully on the CDB and PDBs, and you no longer see `ORA-20001` errors in logs

Related Topics

- *Oracle Database Security Guide*

Oracle Multimedia Downgrade and imrelod.sql Script Error

Review if you encounter ORA-20000: Oracle ORDIM component in registry is status: DOWNGRADED.

When downgrading Oracle Database from release 12.1 to release 11.2.0.2, an error may be raised from the `imrelod.sql` script, which is included with release 11.2.0.2.

```
ORA-20000: Oracle ORDIM component in registry is status: DOWNGRADED.
Oracle ORDIM must be installed and valid prior to Oracle Multimedia
upgrade, downgrade, or patch.
```

- **Cause** The `imrelod.sql` script raises this error because it does not know the status of ORDIM.
- **Action** No action. You can ignore this error.

Oracle Database Vault and dvrelod.sql Script Error

Review if you encounter ORA-31011: XML parsing failed.

When downgrading Oracle Database from release 12.1 to release 11.2.0.3, databases that use Oracle Database Vault may encounter the following error, which is a result of the `dvrelod.sql` script that is included with release 11.2.0.3:

```
ORA-31011: XML parsing failed; Oracle Database Vault downgrade to
release 11.2.0.3
```

- **Cause** The `dvrelod.sql` script does not know the status of XML parsing.

  Action No action. You can ignore this message.

Downgrading Oracle Grid Infrastructure (Oracle Restart) After Successful or Failed Upgrade

To downgrade Oracle Restart, you must deconfigure and then reinstall Oracle Grid Infrastructure. You can then add back the databases and services.

Related Topics

- https://support.oracle.com/rs?type=doc&id=1364412.1
- *Oracle Grid Infrastructure Installation and Upgrade Guide*

Oracle ACFS and Oracle Grid Infrastructure Downgrades to 11g Release 2 (11.2)

You must run `acfsroot install` before you attempt to start the software stack.

If you use Oracle ASM Cluster File System (Oracle ACFS), and you upgrade to Oracle Grid Infrastructure 12c, either for a cluster or for an Oracle Restart standalone server, and you choose to downgrade to Oracle Grid Infrastructure 11g Release 2 (11.2), then before you attempt to start the release 11.2 software stack, you must run the command

`acfsroot install` from the release 11.2 Oracle Grid Infrastructure Oracle home (Grid home).

Database Links Passwords After Downgrading Oracle Database 11*g* Release 1 (11.1)

Reset the passwords for any database links that were created in releases 11.2 or 12.1.

After downgrading to Oracle Database release 11.1, you are required to reset the passwords for any database links that were created in releases 11.2 or 12.1.

- ORA-00600 Errors with Database Links Passwords After Downgrading to Oracle Database 11.1 Release 1
  This error can occur if you do not reset the database link password.
- Using Oracle Data Pump Export to Create a Dump File Containing All Existing Database Links
  Before performing the downgrade, use this Oracle Data Pump export procedure to create a dump file that contains all the existing database links.

ORA-00600 Errors with Database Links Passwords After Downgrading to Oracle Database 11.1 Release 1

This error can occur if you do not reset the database link password.

If you do not reset the database link password, then an internal error is displayed when anyone attempts to make use of the database link. For reference, this is the internal error that is reported in the Oracle trace file when the Oracle server fails to retrieve the password of the database link:

```
ORA-00600: [kzdlk_zt2 err], [18446744073709551601]
```

To reset the password for the database link after downgrading to release 11.1, use the `ALTER DATABASE LINK` command to change the password back to the original password by specifying the original password in the `IDENTIFIED BY` clause.

To create new database links while running Oracle Database release 11.2 or 12.1 that do not have this password issue, contact Oracle support for information about how to use the `IDENTIFIED BY VALUES` clause of the `CREATE DATABASE LINK` command.

Using Oracle Data Pump Export to Create a Dump File Containing All Existing Database Links

Before performing the downgrade, use this Oracle Data Pump export procedure to create a dump file that contains all the existing database links.

The dump file you create includes any newly-created database links. The procedure uses the `FULL=Y` and `INCLUDE=DB_LINK` parameters with the `expdp` command.

1. Log in to SQL*Plus.

For example:

```
sqlplus system/manager
```

2. Drop the dump directory in case one exists.

   For example:

   ```
   SQL> DROP DIRECTORY dpump_dir;
   SQL> CREATE DIRECTORY dpump_dir AS '/
   location_to_write_datapump_dump_file';
   ```

3. Export the database links.

   For example:

   ```
   $ expdp system/manager FULL=Y directory=dpump_dir
     dumpfile=saved_dblinks.dmp INCLUDE=DB_LINK;
   ```

4. After the downgrade, if any of the downgraded database links are not working properly, then drop these links, and import them from the dump file:

   ```
   $ impdp system/manager directory=dpump_dir
   dumpfile=saved_dblinks.dmp;
   ```

   For example, if you find that links are showing the internal error ORA-00600, then dropping and importing the links from the dump file should cause those links to work as intended.

9
Migrating Data Using Oracle Data Pump

Use the Export and Import utilities in Oracle Data Pump to migrate data from one database to another.

Oracle Data Pump provides high performance Export (`expdp`) and Import (`impdp`) utilities. These utilities facilitate upgrading to Oracle Database.

- Overview of Data Pump and Export/Import For Migrating Data
  Oracle provides Oracle Data Pump Export and Import to migrate (move) data from one Oracle Database instance to another.
- Migrating Data With Oracle Data Pump Before Upgrades
  Use this Oracle Data Pump procedure to export data from the source database before you install the new Oracle Database software. Then import the data into the target upgraded database.
- Importing a Full Oracle Database Using a Network Link
  This database export/import method is an alternative to Oracle Data Pump migrations that is helpful when you are migrating to a different storage system.
- Data Pump Requirements When Downgrading Oracle Database
  You can obtain a downward-compatible dump file using Data Pump.

> **See Also:**
>
> *Oracle Database Utilities* for detailed information about Data Pump and the Export and Import utilities

Overview of Data Pump and Export/Import For Migrating Data

Oracle provides Oracle Data Pump Export and Import to migrate (move) data from one Oracle Database instance to another.

Migrating data by using Oracle Data Pump offers the following benefits:

- Supports filtering the metadata that is exported and imported based upon objects and object types, using `INCLUDE` and `EXCLUDE` parameters.
- Supports different modes for unloading/loading portions of the database including: full database mode, schema mode, table mode, tablespace mode, and transportable tablespace mode.
- Enables you to specify how partitioned tables should be handled during import operations, using the `PARTITION_OPTIONS` parameter.
- Provides support for the full range of data types.

`UTL_FIL_DIR` is desupported in Oracle Database 18c. Features using the `UTL_FILE` package to access a symbolic link fails in the new Oracle Database release. To avoid the issue, you must change the directory object and the file name, so that neither contains a symbolic link. This desupport can affect any feature from an earlier release using symbolic links, including (but not restricted to) Oracle Data Pump, BFILEs, and External Tables. If you attempt to use an affected feature after upgrade, where that feature used symbolic links, then you encounter `ORA-29283: invalid file operation: path traverses a symlink`.

Related Topics

- The Export/Import Method for Migrating Data When Upgrading Oracle Database

> **See Also:**
>
> *Oracle Database Utilities* for an overview of Data Pump Export and Import

Migrating Data With Oracle Data Pump Before Upgrades

Use this Oracle Data Pump procedure to export data from the source database before you install the new Oracle Database software. Then import the data into the target upgraded database.

1. Install the new Oracle Database software, and ensure that it is patched to the latest Oracle bundle patch or patch set update (BP or PSU). Installation steps for Oracle Database are covered in your operating system-specific Oracle documentation.
2. Export data from the current database using the Export utility shipped with the current database.

   The current database must not be available for updates during and after the export. If the current database is available to users for updates after the export, then before making the current database available, you must put procedures in place to copy the changes made in the current database to the new database after the import is complete.

   To obtain a consistent point from which you can update the exported database, use one of the following options:

   - Set `FLASHBACK_TIME=SYSTIMESTAMP` in your parameter file, so that the image you obtain of the data in the tables that you export represents the committed state of the table data at the same single point-in-time for all the tables that you are exporting.
   - Use `FLASHBACK_SCN` to select a specific system change number (SCN) that the Export can use to enable the Flashback Query utility.

   Using a flashback option increases UNDO usage and retention.
3. Create the new database. If the new database is on the same server, and it has the same name as the current database, then shut down the current database before creating the new database.

(Option) You can change the storage parameters from the source database. You can pre-create tablespaces, users, and tables in the new 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.

4. Start SQL*Plus in the new Oracle Database environment, and start an Oracle Database instance.

   For example:

   ```
   $ SQLPLUS / AS SYSDBA
   SQL> STARTUP
   ```

5. If you have pre-created items, then specify the `TABLE_EXISTS_ACTION=APPEND` option for Data Pump Import.

   > **Note:**
   >
   > If the new database is created on the same server as the source database, and you do not want to overwrite the source database data files, then you must pre-create the tablespaces and set the following parameter values for the Data pump import:
   >
   > - `REUSE_DATAFILES=N` for Data Pump Import
   >
   >   (Option) You can use the `REMAP_DATAFILE, REMAP_TABLESPACE` and `REMAP_TABLE` options to remap references to export database names in the dump file set to new, non-colliding names in the importing database.
   >
   > - `DESTROY=N` for original Import.

6. Import the objects exported from the current database by using the new database Import utility. To save the informational messages and error messages from the import session to a file, use the following parameters:
   - The `LOGFILE` parameter for Data Pump Import
   - The `LOG` parameter for original Import
7. After the import, check the import log file for information about the imports of specific objects that completed successfully. If there were failures, check for information about any objects that failed.
8. Use further Import scenarios, or use SQL scripts that create the database objects to clean up incomplete imports (or possibly to start an entirely new import).

   > **Note:**
   >
   > If a Data Pump Export or Import job encounters an unrecoverable error, then the job can be restarted after the condition inducing the failure is corrected. The job continues automatically from the point of failure.

9. If changes are made to the current database after the export, then make sure that those changes are propagated to the new database before making it available to users. Refer to step 1 in this procedure.
10. Complete required post-upgrade tasks for your upgrade as described in Chapter 4, “Post-Upgrade Tasks for Oracle Database.”

Related Topics

- *Oracle Database Utilities*

Importing a Full Oracle Database Using a Network Link

This database export/import method is an alternative to Oracle Data Pump migrations that is helpful when you are migrating to a different storage system.

Set up a database link and use the Data Pump Import utility (`impdp`) to perform a full database import from a source database to a destination database. Using this method to migrate data means that dump files are not written, so you do not have to copy over dump files. This method is of particular benefit when you use different storage systems. However, you must work within the limits imposed by the earlier release Oracle Data Pump software during the Oracle Database export.

> **Note:**
>
> To avoid interoperability errors, ensure that you have applied appropriate patchsets to the database you want to upgrade before you start the upgrade.

1. Ensure that the exporting user at the source database has the `DATAPUMP_EXP_FULL_DATABASE` role.

   You must specify this exporting user when you create the database link.
2. Ensure that the importing user at the destination database has the `DATAPUMP_IMP_FULL_DATABASE` role.
3. Create and test a database link between the source and destination databases.
4. Use the following command syntax to start a Data Pump export, where *`import_user`* is the username for the importing user, and *`db_link`* is the name of the database link owned by the exporting user:

   ```
   impdp import_user NETWORK_LINK=db_link FULL=Y NOLOGFILE=Y;
   ```

> **Note:**
>
> Running this command on the importing database implicitly triggers a Data Pump export operation (`expdp`) on the exporting Oracle Database.

5. A log file for the import operation writes to the `DATA_PUMP_DIR` directory. You can discover the location of this directory by running the following command:

   ```
   SQL> select * from dba_directories where DIRECTORY_NAME like
   'DATA_PUMP_DIR';
   ```

   In carrying out this command, be aware that the XDB repository is not moved in a full database export and import operation. LONG columns and NESTED columns are also not moved. However, user-created XML schemas are moved.

> **Note:**
>
> The import operation recreates users on the new destination server. The creation date for users in the `dba_users` table shows the actual import date. The expiration date is updated to the value of the `creation_date` plus the `password_life_time` columns. The creation dates in the `dba_users` table in the destination database are different from the values for the `dba_users` table in the source database.

Related Topics

- Migrating Data With Oracle Data Pump Before Upgrades
  Use this Oracle Data Pump procedure to export data from the source database before you install the new Oracle Database software. Then import the data into the target upgraded database.

> **See Also:**
>
> `http://support.oracle.com` to obtain the latest patchsets

Data Pump Requirements When Downgrading Oracle Database

You can obtain a downward-compatible dump file using Data Pump.

When you use Oracle Data Pump with the downgrade process, the Oracle Database release to which you downgrade must be no more than one release earlier than the release from which you are downgrading.

Use one of the following Data Pump Export methods to obtain a downward-compatible dump file:

- Use the Data Pump Export utility included in the current release Oracle Database home, and set the `VERSION` parameter to the release of the earlier target to which you are downgrading.

  Data Pump Import cannot read dump file sets created by the version of Oracle Data Pump that is later than the current Oracle Database release, unless you

created these dump file sets with the `VERSION` parameter set to the release number of the target database. For this reason, the best way to perform a downgrade is to use Data Pump Export with the `VERSION` parameter set to the release number of the target database to which you are downgrading.

- Use the Data Pump Export utility with DOWNGRADE using the NETWORK_LINK parameter with the VERSION parameter.

  If the compatibility level of the database to which you want to downgrade is earlier than the version of the export dump file sets you created, then you can still transfer data over a database link if the compatibility level of the Oracle Database from which you want to export is within two major release numbers. For example, if one database is Oracle Database 18c, then the other database must be 12*c* release 1 (12.1), or 12*c* release 2 (12.2). You can use Data Pump Export this way to recover from having the VERSION set to an incompatible value during a previous dump file export.

> **Note:**
>
> If you raise the compatibility level after you install the new release of Oracle Database, so that it uses features from the new release, then objects that use those new features cannot be downgraded. For example, if you use the long identifiers available with Oracle Database 18c, then objects with those long identifiers cannot be downgraded. After the downgrade, when you try to import those objects, and the Data Pump Import utility attempts to recreate objects that use long identifiers, you receive an error.

> **See Also:**
>
> *Oracle Database Utilities* for more information about using the `VERSION` parameter

10

Behavior Changes, Deprecated and Desupported Features for Oracle Database

Review for information about Oracle Database 19c changes, deprecations, and desupports, as well as deprecations and desupports in Oracle Database releases 18c, 12.2, and 12.1.

- About Deprecated and Desupported Status
  In addition to new features, Oracle Database releases can modify, deprecate or desupport features, and introduce upgrade behavior changes for your database
- Behavior Changes, Deprecations and Desupports in Oracle Database 19c
  Review for descriptions of Oracle Database 19c release changes.
- Behavior Changes, Deprecations and Desupports in Oracle Database 18c
  Review for descriptions of Oracle Database 18c release changes.
- Behavior Changes, Deprecations and Desupports in Oracle Database 12c Release 2 (12.2)
  Review for descriptions of Oracle Database 12c Release 2 (12.2) changes.
- Behavior Changes, Deprecations and Desupports in Oracle Database 12c Release 1 (12.1)
  Review for descriptions of Oracle Database 12c Release 1 (12.1) changes.

About Deprecated and Desupported Status

In addition to new features, Oracle Database releases can modify, deprecate or desupport features, and introduce upgrade behavior changes for your database

Be aware of the implications of deprecated and desupported:

- **Deprecated** features are features that are no longer being enhanced, but are still supported for the full life of this release of Oracle Database.
- **Desupported** features are features that are no longer supported by fixing bugs related to that feature. Often, Oracle can choose to remove the code required to use the feature. A deprecated feature can be desupported in the next Oracle Database release.

Behavior Changes, Deprecations and Desupports in Oracle Database 19c

Review for descriptions of Oracle Database 19c release changes.

- Behavior Changes for Oracle Database 19c Upgrade Planning
  Review these behavior changes to help plan for upgrades to Oracle Database 19c

- Deprecated Features in Oracle Database 19c
  As part of your upgrade plan, review the features that are deprecated in this Oracle Database release, and review alternatives for your application strategies.
- Deprecated Initialization Parameters in Oracle Database 19c
  As part of your upgrade plan, review the initialization parameters that are deprecated in this Oracle Database release, and review alternatives.
- Desupported Features in Oracle Database 19c
  As part of your upgrade plan, review the desupported features in this Oracle Database release.
- Desupported Parameters in Oracle Database 19c
  As part of your upgrade plan, review the initialization parameters that are not supported starting with this Oracle Database release.

Behavior Changes for Oracle Database 19c Upgrade Planning

Review these behavior changes to help plan for upgrades to Oracle Database 19c

- Changes to Oracle Data Guard Properties Management
  Starting with Oracle Database 19c, properties for Oracle Data Guard configuration are stored in Oracle Database, instead of an external configuration file.
- Rapid Home Provisioning (RHP) Name Change
  Starting with Oracle Database 19c and Oracle Grid Infrastructure 19c, Rapid Home Provisioning is renamed to Fleet Patching and Provisioning (FPP).
- Resupport of Direct File Placement for OCR and Voting Disks
  Starting with Oracle Grid Infrastructure 19c, the desupport for direct placement of OCR and voting files on shared file systems is rescinded for Oracle Standalone Clusters.
- Optional Install for the Grid Infrastructure Management Repository
  Starting with Oracle Grid Infrastructure 19c, the Grid Infrastructure Management Repository (GIMR) is optional for new installations of Oracle Standalone Cluster. Oracle Domain Services Clusters still require the installation of a GIMR as a service component.
- Support for DBMS_JOB
  Oracle continues to support the `DBMS_JOB` package. However, you must grant the `CREATE JOB` privilege to the database schemas that submit `DBMS_JOB` jobs.
- About Standard Edition High Availability
  Starting with Oracle Database 19c Release Update (RU) 19.7, you can install Oracle Database Standard Edition 2 in high availability mode.

Changes to Oracle Data Guard Properties Management

Starting with Oracle Database 19c, properties for Oracle Data Guard configuration are stored in Oracle Database, instead of an external configuration file.

Oracle Data Guard property names, storage locations, and behaviors are changed in Oracle Database 19c.

Property Name Changes

Table 10-1 Oracle Data Guard Property Name Changes

Property	Oracle Database 18c and earlier releases	Oracle Database 19c and later releases
Archive location	`OnlineArchiveLocation`	`ArchiveLocation`
Alternate location	`OnlineAlternateLocation`	`AlternateLocation`
Standby archive location	`StandbyArchiveLocation`	`StandbyArchiveLocation`
Standby alternate location	`StandbyAlternateLocation`	`StandbyAlternateLocation`

Property Behavior Changes

- When `StandbyArchiveLocation` and `StandbyAlternateLocation` have empty strings, `ArchiveLocation` and `AlternateLocation` are locations for both online and standby log files
- When `StandbyArchiveLocation` and `StandbyAlternateLocation` have non-empty strings, `ArchiveLocation` and `AlternateLocation` are locations only for online log files
- The behavior of `StandbyArchiveLocation` and `StandbyAlternateLocation` are not changed. These properties are only used for standby log file locations.

Scope Changes

Starting with Oracle Database 19c, all four of the Oracle Data Guard properties have the scope `Database`. In earlier releases, they had the scope `Instance`.

Imports and Upgrades

Starting with Oracle Database 19c, note the following changes to the way Oracle Data Guard manages property imports and upgrades:

- Oracle Data Guard broker no longer automatically imports local archiving location properties.
- Oracle Data Guard broker no longer automatically upgrades the earlier release property settings from metadata files created from Oracle Database 18c and earlier release Data Guard broker exports.

Rapid Home Provisioning (RHP) Name Change

Starting with Oracle Database 19c and Oracle Grid Infrastructure 19c, Rapid Home Provisioning is renamed to Fleet Patching and Provisioning (FPP).

Resupport of Direct File Placement for OCR and Voting Disks

Starting with Oracle Grid Infrastructure 19c, the desupport for direct placement of OCR and voting files on shared file systems is rescinded for Oracle Standalone Clusters.

In Oracle Grid Infrastructure 12c Release 2 (12.2), Oracle announced that it would no longer support the placement of the Oracle Grid Infrastructure Oracle Cluster Registry (OCR) and voting files directly on a shared file system. This desupport is now rescinded. Starting with Oracle Grid Infrastructure 19c (19.3), with Oracle Standalone Clusters, you can again place OCR and voting disk files directly on shared file systems. However, for Oracle Domain Services Clusters, you must continue to place OCR and voting files in quorum failure groups managed by Oracle Automatic Storage Management (Oracle ASM).

Optional Install for the Grid Infrastructure Management Repository

Starting with Oracle Grid Infrastructure 19c, the Grid Infrastructure Management Repository (GIMR) is optional for new installations of Oracle Standalone Cluster. Oracle Domain Services Clusters still require the installation of a GIMR as a service component.

The Oracle Standalone Cluster locally hosts the GIMR on an Oracle ASM disk group or a shared file system; this GIMR is a multitenant database with a single pluggable database (PDB). The global GIMR runs in an Oracle Domain Services Cluster. Oracle Domain Services Cluster locally hosts the GIMR in a separate Oracle ASM disk group. Client clusters, such as Oracle Member Cluster for Database, use the remote GIMR located on the Oracle Domain Services Cluster. For two-node or four-node clusters, hosting the GIMR for a cluster on a remote cluster reduces the overhead of running an extra infrastructure repository on a cluster. The GIMR for an Oracle Domain Services Cluster is a multitenant database with one PDB, and additional PDB for each member cluster that is added.

Support for DBMS_JOB

Oracle continues to support the `DBMS_JOB` package. However, you must grant the `CREATE JOB` privilege to the database schemas that submit `DBMS_JOB` jobs.

Oracle Scheduler replaces the `DBMS_JOB` package. Although `DBMS_JOB` is still supported for backward compatibility, Oracle strongly recommends that you switch from `DBMS_JOB` to Oracle Scheduler.

In upgrades of Oracle Database 19c and later releases, if the upgrade can recreate existing `DBMS_JOB` jobs using `DBMS_SCHEDULER`, then for backward compatibility, after the upgrade, `DBMS_JOB` continues to act as a legacy interface to the `DBMS_SCHEDULER` job. If existing jobs cannot be recreated using `DBMS_SCHEDULER` because of issues with the metadata, then you receive a `JOB_TABLE_INTEGRITY` warning when you run upgrade prechecks. In that case, you have three options:

- Fix the metadata. After the upgrade continue to run after the upgrade using `DBMS_JOBS` as an interface, and run as `DBMS_SCHEDULER` jobs.
- Drop the jobs, if no longer required.
- Drop `DBMS_JOBS` jobs, and recreate the jobs manually using `DBMS_SCHEDULER`.

For existing jobs created with `DBMS_JOB` that are recreated during the upgrade, the legacy `DBMS_JOB` job is still present as an interface, but using it always creates a `DBMS_SCHEDULER` entry. Apart from the interface, the job is run as a `DBMS_SCHEDULER` job. If you subsequently disable the `DBMS_JOB` job created before the upgrade, then the `DBMS_SCHEDULER` job is also disabled. To avoid this behavior,drop the legacy job, and replace it with a `DBMS_SCHEDULER` job.

For all new jobs, use `DBMS_SCHEDULER`.

About Standard Edition High Availability

Starting with Oracle Database 19c Release Update (RU) 19.7, you can install Oracle Database Standard Edition 2 in high availability mode.

Standard Edition High Availability provides cluster-based failover for Standard Edition Oracle Databases using Oracle Clusterware.

Oracle Standard Edition High Availability benefits from the cluster capabilities and storage solutions that are already part of Oracle Grid Infrastructure, such as Oracle Clusterware, Oracle Automatic Storage Management (Oracle ASM) and Oracle ASM Cluster File System (Oracle ACFS).

Using integrated, shared, and concurrently mounted storage, such as Oracle ASM and Oracle ACFS for database files as well as for unstructured data, enables Oracle Grid Infrastructure to restart an Oracle Database on a failover node much faster than any cluster solution that relies on failing over and remounting volumes and file systems.

Deprecated Features in Oracle Database 19c

As part of your upgrade plan, review the features that are deprecated in this Oracle Database release, and review alternatives for your application strategies.

- Oracle Data Guard Broker Deprecated Properties
  Starting in Oracle Database 19c, several Oracle Data Guard broker properties associated with initialization parameters are deprecated. Their functionality is replaced with the new `EDIT … SET PARAMETER` command in `DGMGRL`.
- Oracle Data Guard Logical Standby Properties Deprecated
  Starting in Oracle Database 19c, Logical Standby properties of Oracle Data Guard broker are deprecated.
- Deprecation of ASMCMD PWCREATE On Command Line
  Using the Oracle ASM command-line utility `ASMCMD` command option `pwcreate` *`password`* to create ASM passwords is deprecated in Oracle Grid Infrastructure 19c (19.1).
- Deprecation of Addnode Script
  The `addnode` script is deprecated in Oracle Grid Infrastructure 19c. The functionality of adding nodes to clusters is available in the installer wizard.
- Deprecation of clone.pl Script
  The `clone.pl` script is deprecated in Oracle Database 19c. The functionality of performing a software-only installation, using the gold image, is available in the installer wizard.
- Deprecation of Oracle Fail Safe
  Oracle Fail Safe is deprecated as of Oracle Database 19c. It can be desupported and unavailable in a future release.
- Deprecation of GDSCTL Operating System Command-Line Password Resets
  To enhance security, starting with Oracle Database 19c, the ability to specify passwords from the Global Data Services Control Utility (GDSCTL) command-line when called from the operating system prompt is deprecated.

- Deprecation of Oracle Enterprise Manager Express
  Flash-based Enterprise Manager Express is deprecated in Oracle Database 19c. Starting with Oracle Database 19c, Enterprise Manager Express uses Java JET technology for the user interface.
- Deprecation of DV_REALM_OWNER Role
  The Oracle Data Vault role `DV_REALM_OWNER` role is deprecated with no replacement.
- Deprecation of DV_REALM_RESOURCE Role
  The Oracle Data Vault role `DV_REALM_RESOURCE` is deprecated with no replacement.
- Deprecation of DV_PUBLIC Role
  The Oracle Data Vault role `DV_PUBLIC` role is deprecated with no replacement.
- Deprecation of Oracle ACFS Replication Protocol REPV1
  Starting with Oracle Database 19c (19.3), the Oracle ACFS replication protocol repv1 is deprecated.
- Deprecation of Oracle ACFS Encryption on Solaris and Windows
  Starting with Oracle Database 19c (19.3), Oracle ACFS Encryption is deprecated with no replacement on Oracle Solaris and Microsoft Windows.
- Deprecation of Oracle ACFS on Windows
  Starting with Oracle Grid Infrastructure 19c (19.5), Oracle ASM Cluster File System (ACFS) is deprecated on Microsoft Windows.
- Deprecation of Oracle ACFS Security (Vault) and ACFS Auditing
  Starting with Oracle Grid Infrastructure 19c (19.5), Oracle ASM Cluster File System (ACFS) Security (Vault) and ACFS Auditing are deprecated
- Deprecation of Oracle ACFS on Member Clusters (ACFS Remote)
  Starting with Oracle Grid Infrastructure 19c (19.5), Oracle ASM Cluster File System (ACFS) on Member Clusters (ACFS Remote) is deprecated.
- Deprecation of Cluster Domain - Member Clusters
  Starting with Oracle Grid Infrastructure 19c (19.5), Member Clusters, which are part of the Oracle Cluster Domain architecture, are deprecated.
- Deprecation of Vendor Clusterware Integration with Oracle Clusterware
  Starting with Oracle Clusterware 19c (19.5), the integration of vendor or third party clusterware with Oracle Clusterware is deprecated.
- Deprecation of Black Box Virtual Machine Management Using Oracle Clusterware
  Direct management of virtual machine (VM) resources using Oracle Clusterware is deprecated in Oracle Grid Infrastructure 19c, and can be removed in a future release.

Oracle Data Guard Broker Deprecated Properties

Starting in Oracle Database 19c, several Oracle Data Guard broker properties associated with initialization parameters are deprecated. Their functionality is replaced with the new `EDIT … SET PARAMETER` command in `DGMGRL`.

The following Oracle Data Guard broker properties are deprecated in Oracle Database 19c:

- `ArchiveLagTarget`
- `DataGuardSyncLatency`
- `LogArchiveMaxProcesses`

- LogArchiveMinSucceedDest
- LogArchiveTrace
- StandbyFileManagement
- DbFileNameConvert
- LogArchiveFormat
- LogFileNameConvert

Using the current EDIT ... SET PROPERTY command with these properties continues to work. However, the update is automatically made with the new command, and the parameter data is now no longer be stored in the broker metadata file.

The InconsistentProperties property is also deprecated. This parameter now always has no value, because there can no longer be inconsistent values.

Using the new EDIT ... SET PARAMETER commands removes the possibility of inconsistent configuration data between the broker and a database. When you use the new EDIT...SET PARAMETER commands, you can change these parameters either by using either the new broker command, or by using the standard SQL*Plus ALTER SYSTEM command. However, when you use the broker command, you can be attached to any database in the configuration, and perform parameter changes to any other database in the configuration.

Oracle Data Guard Logical Standby Properties Deprecated

Starting in Oracle Database 19c, Logical Standby properties of Oracle Data Guard broker are deprecated.

The following Oracle Data Guard broker Properties that affect Logical Standby are deprecated:

- LsbyMaxEventsRecorded
- LsbyMaxServers
- LsbyMaxSga
- LsbyPreserveCommitOrder
- LsbyRecordAppliedDdl
- LsbyRecordSkippedDdl
- LsbyRecordSkipErrors
- LsbyParameter

Using the EDIT ... SET PROPERTY command continues to work. However, the data about the setting is no longer stored in the broker metadata file. Instead, Oracle recommends that you use the SQL*Plus package DBMS_LOGSTDBY to change the Logical Standby properties. The Logical Standby properties for Oracle Data Guard broker can be desupported in a future release.

Directly using the SQL*Plus package DBMS_LOGSTDBY removes the possibility of inconsistent configuration data between the broker and a Logical Standby database, and provides one interface to manage a Logical Standby.

Deprecation of ASMCMD PWCREATE On Command Line

Using the Oracle ASM command-line utility `ASMCMD` command option `pwcreate` *`password`* to create ASM passwords is deprecated in Oracle Grid Infrastructure 19c (19.1).

The option to supply the password on the command line is still enabled in Oracle Database 19c. However, to enhance security, Oracle is deprecating this method of creating a new Oracle ASM password. It can be desupported in a future release. The `pwcreate` option of `ASMCMD` enables you to specify a password on the command line. However, if you run the command `asmcmd pwcreate`, and you do not provide the password on the command line, then you are now prompted for the password.

Deprecation of Addnode Script

The `addnode` script is deprecated in Oracle Grid Infrastructure 19c. The functionality of adding nodes to clusters is available in the installer wizard.

The `addnode` script can be removed in a future release. Instead of using the `addnode` script (`addnode.sh` or `addnode.bat`), add nodes by using the installer wizard. The installer wizard provides many enhancements over the `addnode` script. Using the installer wizard simplifies management by consolidating all software lifecycle operations into a single tool.

Deprecation of clone.pl Script

The `clone.pl` script is deprecated in Oracle Database 19c. The functionality of performing a software-only installation, using the gold image, is available in the installer wizard.

The `clone.pl` script can be removed in a future release. Instead of using the `clone.pl` script, Oracle recommends that you install the extracted gold image as a home, using the installer wizard.

Deprecation of Oracle Fail Safe

Oracle Fail Safe is deprecated as of Oracle Database 19c. It can be desupported and unavailable in a future release.

Oracle recommends that you evaluate other single-node failover options, such as Oracle RAC One Node.

Deprecation of GDSCTL Operating System Command-Line Password Resets

To enhance security, starting with Oracle Database 19c, the ability to specify passwords from the Global Data Services Control Utility (GDSCTL) command-line when called from the operating system prompt is deprecated.

This deprecation applies only to password changes where GDSCTL is called from a user command-line prompt. For example, the following command is deprecated.

```
$ gdsctl add database -connect inst1 -pwd gsm_password
```

Specifying the password from the `GDSCTL` utility itself is still valid. For example, the following command is valid:

```
GDSCTL> add database -connect inst1 -pwd gsm_password
```

This deprecation addresses the security vulnerability when specifying passwords in GDSCTL commands called from the operating system prompt. Only enter the Global Data Services password only when GDSCTL prompts for it.

Deprecation of Oracle Enterprise Manager Express

Flash-based Enterprise Manager Express is deprecated in Oracle Database 19c. Starting with Oracle Database 19c, Enterprise Manager Express uses Java JET technology for the user interface.

In accordance with industry standards, Oracle is deprecating Flash-based Oracle Enterprise Manager Express (Oracle EM Express). Starting with Oracle Database 19c, Oracle EM Express, the default management option for Oracle Database, is based on Java JET technology. In this initial release, there are some options available in Flash-based Oracle EM Express that are not available in the JET version. If necessary, use the following command to revert to Flash Oracle EM Express:

```
SQL> @?/rdbms/admin/execemx emx
```

To return to JET Oracle EM Express, use the following command:

```
SQL> @?/rdbms/admin/execemx omx
```

Deprecation of DV_REALM_OWNER Role

The Oracle Data Vault role `DV_REALM_OWNER` role is deprecated with no replacement.

The `DV_REALM_OWNER` role is used for realm management to manage database objects in multiple schemas that define a realm. Oracle has deprecated the use of this role. It can be removed in a future release.

In addition, the following DV_REALM_OWNER privileges are revoked from the `DV_REALM_OWNER` role: `CREATE ROLE`, `ALTER ANY ROLE`, `DROP ANY ROLE`, `GRANT ANY ROLE`, `GRANT ANY PRIVILEGE`, and `GRANT ANY OBJECT PRIVILEGE`. If needed, you can choose to grant these privileges to the `DV_REALM_OWNER` role. For example:

```
SQL> GRANT CREATE ROLE ON tablename TO DV_REALM_OWNER;
```

Deprecation of DV_REALM_RESOURCE Role

The Oracle Data Vault role `DV_REALM_RESOURCE` is deprecated with no replacement.

The `DV_REALM_RESOURCE` role is used for the management of realm resources. Oracle has deprecated the use of this role. It can be removed in a future release.

Deprecation of DV_PUBLIC Role

The Oracle Data Vault role `DV_PUBLIC` role is deprecated with no replacement.

The DV_PUBLIC role is still created during installation, but it is not granted any roles or privileges. All privileges that were granted to DV_PUBLIC in previous releases are

now granted directly to the PUBLIC role. This role is obsolete, and can be removed in a future release.

Deprecation of Oracle ACFS Replication Protocol REPV1

Starting with Oracle Database 19c (19.3), the Oracle ACFS replication protocol repv1 is deprecated.

The initial ACFS replication protocol `repv1` was released with Oracle Database 11g Release 2 (11.2). Oracle Database 12c Release 2 introduced a new ACFS replication protocol, Oracle ACFS snapshot-based replication (`repv2`). Oracle continued to use the same management interface. Starting with Oracle Database 19c, the earlier ACFS replication protocol (`repv1`) is deprecated. Update to snapshot-based replication.

Deprecation of Oracle ACFS Encryption on Solaris and Windows

Starting with Oracle Database 19c (19.3), Oracle ACFS Encryption is deprecated with no replacement on Oracle Solaris and Microsoft Windows.

Oracle ACFS Encryption on Oracle Solaris and Microsoft Windows is based on RSA technology. Retirement of RSA technology has been announced. Oracle ACFS Encryption continues to be supported on Linux, and is unaffected by this deprecation, because Linux uses an alternative technology.

Deprecation of Oracle ACFS on Windows

Starting with Oracle Grid Infrastructure 19c (19.5), Oracle ASM Cluster File System (ACFS) is deprecated on Microsoft Windows.

Deprecating certain clustering features with limited adoption allows Oracle to focus on improving core scaling, availability, and manageability across all features and functionality. ACFS file systems on Microsoft Windows are deprecated, and can be desupported in a future release. Depending on the use case, to replace current ACFS file systems, Oracle recommends that you move to Oracle Automatic Storage Management (Oracle ASM), Oracle Database File System (DBFS), or Microsoft Windows shares.

Deprecation of Oracle ACFS Security (Vault) and ACFS Auditing

Starting with Oracle Grid Infrastructure 19c (19.5), Oracle ASM Cluster File System (ACFS) Security (Vault) and ACFS Auditing are deprecated

Deprecating certain clustering features with limited adoption allows Oracle to focus on improving core scaling, availability, and manageability across all features and functionality. Oracle ACFS Security (Vault) and ACFS Auditing are deprecated, and can be desupported in a future release.

Deprecation of Oracle ACFS on Member Clusters (ACFS Remote)

Starting with Oracle Grid Infrastructure 19c (19.5), Oracle ASM Cluster File System (ACFS) on Member Clusters (ACFS Remote) is deprecated.

Oracle ASM Cluster File System (ACFS) on Member Clusters (ACFS Remote) is deprecated, and can be removed in a future release. Deprecating certain clustering

features with limited adoption allows Oracle to focus on improving core scaling, availability, and manageability across all features and functionality.

Deprecation of Cluster Domain - Member Clusters

Starting with Oracle Grid Infrastructure 19c (19.5), Member Clusters, which are part of the Oracle Cluster Domain architecture, are deprecated.

Deprecating certain clustering features with limited adoption allows Oracle to focus on improving core scaling, availability, and manageability across all features and functionality. Oracle Cluster Domains consist of a Domain Services Cluster (DSC) and Member Clusters. The deprecation of Member Clusters affects the clustering used with the DSC, but not its ability to host services for other production clusters. Oracle recommends that you align your next software or hardware upgrade with the transition off Cluster Domain - Member Clusters.

Deprecation of Vendor Clusterware Integration with Oracle Clusterware

Starting with Oracle Clusterware 19c (19.5), the integration of vendor or third party clusterware with Oracle Clusterware is deprecated.

The integration of vendor clusterware with Oracle Clusterware is deprecated, and can be desupported in a future release. Deprecating certain clustering features with limited adoption allows Oracle to focus on improving core scaling, availability, and manageability across all features and functionality. In the absence of an integration between different cluster solutions, the system is subject to the dueling cluster solutions issue: Independent cluster solutions can make individual decisions about which corrective actions must be taken in case of certain failures. To avoid conflicts, only one cluster solution should be active at any point in time. For this reason, Oracle recommends that you align your next software or hardware upgrade with the transition off of vendor cluster solutions.

Deprecation of Black Box Virtual Machine Management Using Oracle Clusterware

Direct management of virtual machine (VM) resources using Oracle Clusterware is deprecated in Oracle Grid Infrastructure 19c, and can be removed in a future release.

Oracle continues to support the use of Oracle Clusterware to monitor and manage the availability of remotely deployed Oracle VMs. However, this support will be removed in an upcoming release.

Deprecated Initialization Parameters in Oracle Database 19c

As part of your upgrade plan, review the initialization parameters that are deprecated in this Oracle Database release, and review alternatives.

- CLUSTER_DATABASE_INSTANCES Initialization Parameter Deprecated
  The Oracle Database initialization parameter `CLUSTER_DATABASE_INSTANCES` is deprecated in Oracle Database 19c (19.1)
- Deprecation of SQLNET.ENCRYPTION_WALLET_LOCATION Parameter
  The SQLNET.ENCRYPTION_WALLET_LOCATION `sqlnet.ora` parameter is deprecated in Oracle Database 19c.

- Deprecation of the SERVICE_NAMES Initialization Parameter
  Starting with Oracle Database 19c, customer use of the SERVICE_NAMES parameter is deprecated. It can be desupported in a future release.

CLUSTER_DATABASE_INSTANCES Initialization Parameter Deprecated

The Oracle Database initialization parameter CLUSTER_DATABASE_INSTANCES is deprecated in Oracle Database 19c (19.1)

The init.ora parameter CLUSTER_DATABASE_INSTANCES specifies the number of configured Oracle Real Application Clusters (Oracle RAC) instances. Starting with Oracle Database 19c and later releases, the number of configurable Oracle RAC instances is derived automatically from the Oracle Clusterware resource definitions. There is no replacement for this parameter, because there is no longer a reason to have this parameter.

Deprecation of SQLNET.ENCRYPTION_WALLET_LOCATION Parameter

The SQLNET.ENCRYPTION_WALLET_LOCATION sqlnet.ora parameter is deprecated in Oracle Database 19c.

The SQLNET.ENCRYPTION_WALLET_LOCATION parameter defines the location of the software keystores for Transparent Data Encryption (TDE). To configure the software keystore location, instead of setting SQLNET.ENCRYPTION_WALLET_LOCATION, Oracle recommends that you set the WALLET_ROOT initialization parameter, and the TDE_CONFIGURATION dynamic initialization parameter.

Oracle recommends that you use the WALLET_ROOT instance initialization parameter as soon as possible, because the value is read once at instance startup time, so all sessions and server background processes share the same path after startup. If the SQLNET.ENCRYPTION_WALLET_LOCATION parameter is used, then it can lead to confusing misconfigurations, because different sessions can have different SQLNET parameter values. Another reason to use WALLET_ROOT is that it is the directory within which you can locate the wallets of other features, such as Oracle Enterprise User Security, and Transport Layer Security. This location can become the principal location for all server-side wallets.

Related Topics

- *Oracle Database Advanced Security Guide*

Deprecation of the SERVICE_NAMES Initialization Parameter

Starting with Oracle Database 19c, customer use of the SERVICE_NAMES parameter is deprecated. It can be desupported in a future release.

The use of the SERVICE_NAMES parameter is no longer actively supported. It must not be used for high availability (HA) deployments. It is not supported to use service names parameter for any HA operations. This restriction includes FAN, load balancing, FAILOVER_TYPE, FAILOVER_RESTORE, SESSION_STATE_CONSISTENCY, and any other uses.

To manage your services, Oracle recommends that you use the SRVCTL or GDSCTL command line utilities, or the DBMS_SERVICE package.

Desupported Features in Oracle Database 19c

As part of your upgrade plan, review the desupported features in this Oracle Database release.

- Desupport of Oracle Data Provider for .NET Promotable Transaction Setting
  The Oracle Data Provider for .NET `PromotableTransaction` setting is desupported because it is no longer necessary. All compatible database server versions support transaction promotion.
- Desupport of Oracle Multimedia
  Oracle Multimedia is desupported in Oracle Database 19c, and the implementation is removed.
- Desupport of the CONTINUOUS_MINE feature of LogMiner
  The `continuous_mine` option for the `dbms_logmnr.start_logmnr` package is desupported in Oracle Database 19c, and is no longer available.
- Desupport of Extended Datatype Support (EDS)
  The Extended Datatype Support (EDS) feature is desupported in Oracle Database 19c. All Data types that the EDS feature supported are now supported natively by both Logical Standby and Oracle GoldenGate.
- Data Guard Broker MaxConnections Property Desupported
  Starting in Oracle Database 19c, the Oracle Data Guard broker `MAX_CONNECTIONS` attribute is desupported.
- Desupport of Leaf Nodes in Flex Cluster Architecture
  Leaf nodes are no longer supported in the Oracle Flex Cluster Architecture in Oracle Grid Infrastructure 19c.
- Desupport of Oracle Streams
  Starting in Oracle Database 19c (19.1), Oracle Streams is desupported. Oracle GoldenGate is the replication solution for Oracle Database.
- Desupport of PRODUCT_USER_PROFILE Table
  Starting in Oracle Database 19c, the SQL*Plus table `PRODUCT_USER_PROFILE` (PUP table) is desupported.
- Desupport of Oracle Real Application Clusters for Standard Edition 2 (SE2) Database Edition
  Starting with Oracle Database 19c, Oracle Real Application Clusters (Oracle RAC) is not supported in Oracle Database Standard Edition 2 (SE2).
- Desupport of PDB Flat File Dictionary Dumps
  The ability to create flat file dictionary dumps of pluggable databases (PDBs) is desupported in Oracle Database 19c.

Desupport of Oracle Data Provider for .NET Promotable Transaction Setting

The Oracle Data Provider for .NET `PromotableTransaction` setting is desupported because it is no longer necessary. All compatible database server versions support transaction promotion.

The Oracle Data Provider for .NET registry, configuration, and property setting `PromotableTransaction` indicates whether the application must keep transactions as local, or if it can begin all single connection transactions as local, and then promote

the transaction to distributed when a second connection enlists. This is the concept of promotable transactions.

The `PromotableTransaction` setting is desupported in Oracle Data Provider for .NET 18c, because all database versions compatible with this provider version support promotable transactions. Developers no longer need to use this setting if they employ promotable transactions. Existing applications remain unaffected, whether they use promotable transactions or not.

Desupport of Oracle Multimedia

Oracle Multimedia is desupported in Oracle Database 19c, and the implementation is removed.

As an alternative for image processing and conversion, Oracle recommends that you store multimedia content in SecureFiles LOBs, and use third party products, such as APEX Media Extension (AME). The `ORDIM` component remains in the registry and still has a `VALID` status. Oracle Multimedia objects and packages remain in the database. However, these objects and packages no longer function, and raise exceptions if there is an attempt made to use them. Oracle Locator is not affected by the desupport of Oracle Multimedia.

Related Topics

- https://www.apexmediaextension.com

Desupport of the CONTINUOUS_MINE feature of LogMiner

The `continuous_mine` option for the `dbms_logmnr.start_logmnr` package is desupported in Oracle Database 19c, and is no longer available.

The `continuous_mine` functionality of the LogMiner package is obsolete. It was deprecated in Oracle Database 12c Release 2 (12.2). There is no replacement functionality.

Desupport of Extended Datatype Support (EDS)

The Extended Datatype Support (EDS) feature is desupported in Oracle Database 19c. All Data types that the EDS feature supported are now supported natively by both Logical Standby and Oracle GoldenGate.

The Extended Datatype Support (EDS) feature provides a mechanism for logical standbys to support certain Oracle data types that lack native redo-based support. For example, EDS was used to replicate tables with a `SDO_GEOMETRY` column. However, starting with Oracle Database 12c Release 2 (12.2), there are no EDS-supported Oracle data types that are not supported natively, either by Logical standby, or by Oracle GoldenGate. This feature is desupported with Oracle Database 19c (19.1).

Data Guard Broker MaxConnections Property Desupported

Starting in Oracle Database 19c, the Oracle Data Guard broker `MAX_CONNECTIONS` attribute is desupported.

The Oracle Data Guard broker `MaxConnections` property (pertaining to the `MAX_CONNECTIONS` attribute of the `LOG_ARCHIVE_DEST_`*n* parameter) is desupported in

Oracle Database 19c. It is removed. Using commands to set this property from DGMGRL returns errors.

Desupport of Leaf Nodes in Flex Cluster Architecture

Leaf nodes are no longer supported in the Oracle Flex Cluster Architecture in Oracle Grid Infrastructure 19c.

In Oracle Grid Infrastructure 19c (19.1) and later releases, all nodes in an Oracle Flex Cluster function as hub nodes. The capabilities offered by Leaf nodes in the original implementation of the Oracle Flex Cluster architecture can as easily be served by hub nodes. Therefore, leaf nodes are no longer supported.

Desupport of Oracle Streams

Starting in Oracle Database 19c (19.1), Oracle Streams is desupported. Oracle GoldenGate is the replication solution for Oracle Database.

Note that Oracle Database Advanced Queuing is not deprecated, and is fully supported in Oracle Database 19c. Oracle Streams did not support features added in Oracle Database 12c (12.1) and later releases, including the multitenant architecture, `LONG VARCHAR`, and other new features. Oracle Streams replication functionality is superseded by GoldenGate.

Desupport of PRODUCT_USER_PROFILE Table

Starting in Oracle Database 19c, the SQL*Plus table `PRODUCT_USER_PROFILE` (PUP table) is desupported.

The SQL*Plus product-level security feature is unavailable in Oracle Database 19c. Oracle recommends that you protect data by using Oracle Database settings, so that you ensure consistent security across all client applications.

Desupport of Oracle Real Application Clusters for Standard Edition 2 (SE2) Database Edition

Starting with Oracle Database 19c, Oracle Real Application Clusters (Oracle RAC) is not supported in Oracle Database Standard Edition 2 (SE2).

Upgrading Oracle Database Standard Edition databases that use Oracle Real Application Clusters (Oracle RAC) functionality from earlier releases to Oracle Database 19c is not possible. To upgrade those databases to Oracle Database 19c, either remove the Oracle RAC functionality before starting the upgrade, or upgrade from Oracle Database Standard Edition to Oracle Database Enterprise Edition. For more information about each step, including how to reconfigure your system after an upgrade, see My Oracle Support Note 2504078.1: "Desupport of Oracle Real Application Clusters (RAC) with Oracle Database Standard Edition 19c."

Related Topics

- My Oracle Support Document 2504078.1

Desupport of PDB Flat File Dictionary Dumps

The ability to create flat file dictionary dumps of pluggable databases (PDBs) is desupported in Oracle Database 19c.

In previous releases, using a flat file dictionary was one means of mining the redo logs for the changes associated with a specific PDB whose data dictionary was contained within the flat file. This feature is now desupported. With Oracle Database 19c and later releases, Oracle recommends that you call `DBMS_LOGMNR.START_LOGMNR`, and supply the system change number (SCN) or time range that you want to mine. The SCN or time range options of `START_LOGMNR` are enhanced to support mining of individual PDBs.

Desupported Parameters in Oracle Database 19c

As part of your upgrade plan, review the initialization parameters that are not supported starting with this Oracle Database release.

- EXAFUSION_ENABLED Initialization Parameter Desupported
  The Oracle Exadata Database Machine initialization parameter `EXAFUSION_ENABLED` is desupported in Oracle Database 19c.
- MAX_CONNECTIONS attribute of LOG_ARCHIVE_DEST_n Desupported
  The `MAX_CONNECTIONS` attribute of the `LOG_ARCHIVE_DEST_`*n* parameters for Oracle Data Guard Redo Transport is obsolete. It is desupported in Oracle Database 19c.
- Desupport of O7_DICTIONARY_ACCESS
  The initialization parameter `O7_DICTIONARY_ACCESSIBILITY` is desupported in Oracle Database 19c.
- Desupport of OPTIMIZE_PROGRESS_TABLE Parameter
  `OPTIMIZE_PROGRESS_TABLE` for Oracle GoldenGate Integrated Replicat, XStream In, and Logical Standby, is desupported in Oracle Database 19c.

EXAFUSION_ENABLED Initialization Parameter Desupported

The Oracle Exadata Database Machine initialization parameter `EXAFUSION_ENABLED` is desupported in Oracle Database 19c.

The Exafusion feature was introduced for Oracle Database 12c Release 1 (12.1.0.2), but disabled by default. It was only available for the Linux operating system, and only available with Oracle Exadata Database Machine. You could enable this feature by setting the `EXAFUSION_ENABLED` initialization parameter to `1`. With Oracle Database 12c Release 2 (12.2), the feature became enabled by default on Oracle Exadata Database Machine running on Oracle Linux. You could disable this feature by changing the `EXAFUSION_ENABLED` parameter setting to `0`. However, with Oracle Database 18c and later releases, the Exafusion feature cannot be disabled. For this reason, the `EXAFUSION_ENABLED` parameter is desupported in Oracle Database 19c, because the parameter no longer serves a function.

MAX_CONNECTIONS attribute of LOG_ARCHIVE_DEST_*n* Desupported

The `MAX_CONNECTIONS` attribute of the `LOG_ARCHIVE_DEST_`*n* parameters for Oracle Data Guard Redo Transport is obsolete. It is desupported in Oracle Database 19c.

The MAX_CONNECTIONS attribute can interfere with the new Redo Transport Streaming mechanism introduced in Oracle Database 11g, and increase the time necessary to resolve gaps. To prevent these types of errors, Oracle has desupported and removed this attribute.

Desupport of O7_DICTIONARY_ACCESS

The initialization parameter O7_DICTIONARY_ACCESSIBILITY is desupported in Oracle Database 19c.

The O7_DICTIONARY_ACCESSIBILITY parameter controlled restrictions on System Privileges from accessing SYS owned objects. It was retained to enable certain backward compatibility for earlier release applications. Desupporting obsolete features enables Oracle to focus on security across all features and functionality. Oracle recommends that you manage system privileges in accordance with standard security best practices.

Desupport of OPTIMIZE_PROGRESS_TABLE Parameter

OPTIMIZE_PROGRESS_TABLE for Oracle GoldenGate Integrated Replicat, XStream In, and Logical Standby, is desupported in Oracle Database 19c.

The apply parameter OPTIMIZE_PROGRESS_TABLE for Oracle GoldenGate Integrated Replicat, XStream In, and Logical Standby, is desupported in Oracle Database 19c. Before you upgrade to Oracle Database 19, you must turn off this parameter. If OPTIMIZE_PROGRESS_TABLE is set to ON, then stop apply gracefully, turn off the parameter, and restart apply. For GoldenGate Apply and XStream, this parameter is set to OFF by default.

Behavior Changes, Deprecations and Desupports in Oracle Database 18c

Review for descriptions of Oracle Database 18c release changes.

- Behavior Changes for Oracle Database 18c Upgrade Planning
  Review these behavior changes to help plan for upgrades to Oracle Database 18c
- Deprecated Features in Oracle Database 18c
  Review the deprecated features listed in this section to prepare to use alternatives after you upgrade.
- Desupported Features in Oracle Database 18c
  Review this list of desupported features as part of your upgrade planning.
- Desupported Initialization Parameters in Oracle Database 18c
  Review this list of desupported initialization parameters for changes and replacements in parameter settings in this release.
- Deprecation of Oracle Multimedia
  Starting in Oracle Database 18c, Oracle Multimedia is deprecated. Oracle Multimedia will be desupported in Oracle Database 19c.
- Terminal Release of Oracle Streams
  Oracle Database 18c is the terminal release for Oracle Streams support. Oracle Streams will be desupported from Oracle Database 19c onwards.

Behavior Changes for Oracle Database 18c Upgrade Planning

Review these behavior changes to help plan for upgrades to Oracle Database 18c

- Simplified Image-Based Oracle Database Installation
  Starting with Oracle Database 18c, installation and configuration of Oracle Database software is simplified with image-based installation.
- Support Indexing of JSON Key Names Longer Than 64 Characters
  If you use JSON keys, then you can take advantage of increased efficiency of searching JSON documents generated from HASH MAP-like structures by using longer key names.
- Upgrading Existing Databases is Replaced With Image Installations
  Starting with Oracle Database 18c, existing services are no longer migrated by the installation. Use Database Upgrade Assistant (DBUA) to migrate services.
- About RPM-Based Oracle Database Installation
  Starting with Oracle Database 18c, you can install a single-instance Oracle Database or an Oracle Database Instant Client software using RPM packages.
- Token Limitations for Oracle Text Indexes
  Starting with Oracle Database Release 18c, the indexed token maximum size is increased to 255 characters for single-byte character sets.
- Changes to /ALL/USER/DBA User View and PL/SQL External Libraries
  Starting in Oracle Database 18c, there are changes to the `/USER/ALL/DBA_ARGUMENTS` and `/USER/ALL/DBA_IDENTIFIERS` views, and to `LIBRARY` object creation in PDBs.
- Symbolic Links and UTL_FILE
  You cannot use `UTL_FILE` with symbolic links. Use directory objects instead.
- Deprecation of Direct Registration of Listeners with DBCA
  Using Database Configuration Assistant (DBCA) to register Oracle Database to Oracle Internet Directory (OID) is deprecated in Oracle Database 18c.
- UNIFORM_LOG_TIMESTAMP_FORMAT Changes in INIT.ORA
  By default, the format of timestamps is different in Oracle Database 12*c* release 2 (12.2) and later releases. To view alert logs, use the Oracle Database utility Automatic Diagnostic Repository Command Interpreter (ADRCI) utility.

Simplified Image-Based Oracle Database Installation

Starting with Oracle Database 18c, installation and configuration of Oracle Database software is simplified with image-based installation.

Starting with Oracle Database 18c, the Oracle Database software is available as an image file for download and installation. You must extract the image software into the directory where you want your Oracle home to be located, and then run the `runInstaller` script to start the Oracle Database installation. For details, refer to your operating system platform *Oracle Database Installation Guide*.

> **Note:**
>
> You must extract the image software (`db_home.zip`) into the directory where you want your Oracle Database home to be located, and then run the `runInstaller` script to start the Oracle Database installation and configuration. Oracle recommends that the Oracle home directory path you create is in compliance with the Oracle Optimal Flexible Architecture recommendations.

Related Topics

- *Oracle Database Installation Guide*

Support Indexing of JSON Key Names Longer Than 64 Characters

If you use JSON keys, then you can take advantage of increased efficiency of searching JSON documents generated from HASH MAP-like structures by using longer key names.

The upper limit is increased for JSON key names that can be indexed by the JSON Search index. The JSON key name upper limit in Oracle Database 12*c* Release 2 (12.2.0.2) and later releases is 255 bytes. In previous releases, JSON search indexes that were created did not index key names greater than 64 bytes.

Upgrading Existing Databases is Replaced With Image Installations

Starting with Oracle Database 18c, existing services are no longer migrated by the installation. Use Database Upgrade Assistant (DBUA) to migrate services.

If you have an existing Oracle Database with services that you want to migrate, then to migrate those services, you must install the new release Oracle Database software in the Oracle home, and then start DBUA.

On Windows, to migrate the Microsoft Transaction Service to the new Oracle home, you must also run the command `%ORACLE_HOME%\bin\oramtsctl.exe -new`

About RPM-Based Oracle Database Installation

Starting with Oracle Database 18c, you can install a single-instance Oracle Database or an Oracle Database Instant Client software using RPM packages.

An RPM-based installation performs preinstallation checks, extracts the database software, reassigns ownership of the extracted software to the preconfigured user and groups, maintains the Oracle inventory, and executes all root operations required to configure the Oracle Database software for a single-instance Oracle Database creation and configuration.

The RPM–based installation process detects when the minimum requirements for an installation are not met and prompts you to finish these minimum preinstallation requirements.

An RPM-based installation performs a software-only Oracle Database installation and creates an Oracle home. After the Oracle home is created, you can then use Oracle Database Configuration Assistant (Oracle DBCA) to create an Oracle Database.

The RPM-based installation process provides you with the option to create a database with the default settings using the `/etc/init.d/oracledb_ORCLCDB-19c` service configuration script.

Token Limitations for Oracle Text Indexes

Starting with Oracle Database Release 18c, the indexed token maximum size is increased to 255 characters for single-byte character sets.

Before Oracle Database Release 18c, all Oracle Text index types except `SDATA` sections stored tokens in a table column of type `VARCHAR2` (64 BYTE). Starting with Oracle Database Release 18c, all Oracle Text index types except `CTXCAT` and `CTXRULE` indexes store tokens in `VARCHAR2` (255 BYTE) table column types. This change is an increase for the maximum size of an indexed token to 255 characters for single-byte character sets. The size increase is less with multibyte or variable-length character sets. Tokens longer than 255 bytes are truncated. Truncated tokens do not prevent searches on the whole token string. However, the system cannot distinguish between two tokens that have the same first 255 bytes.

> **Note:**
>
> Before Oracle Database Release 18c, tokens that were greater than 64 bytes were truncated to 64 bytes. After upgrading to Oracle Database Release 18c, the token tables are increased to 255 bytes from 64 bytes. Searches with more than 64 bytes in the search token (that is, any single word in search string) cannot find any tokens which were truncated to 64 bytes. To avoid this problem, rebuild the index. If you never use search tokens longer than 64 bytes, it is not necessary to rebuild the index.

`SDATA` sections store tokens in a table column of type `VARCHAR2` (249 BYTE). `CTXCAT` and `CTXRULE` indexes store tokens in a table column of type `VARCHAR2` (64 BYTE).

Changes to /ALL/USER/DBA User View and PL/SQL External Libraries

Starting in Oracle Database 18c, there are changes to the `/USER/ALL/DBA_ARGUMENTS` and `/USER/ALL/DBA_IDENTIFIERS` views, and to `LIBRARY` object creation in PDBs.

Review the changes that can affect your work.

ALL/USER/DBA_ARGUMENTS User Views Changes

`ARGUMENTS` views contain fewer rows. In particular, only top-level (`DATA_LEVEL=0`) items are stored in the `ARGUMENTS` views.

In earlier Oracle Database releases, the PL/SQL compiler collected metadata for all nested types in a PL/SQL datatype. `DATA_LEVEL` represented the nesting level of the type. Starting in Oracle Database 18c, only top-level type metadata (`DATA_LEVEL=0`) is stored in the `ARGUMENTS` views.

For instance: Note the changes in the create-or-replace package `NestedTypesExample`:

```
Type Level2Record is RECORD (Field1 NUMBER);
Type Level1Collection is TABLE of Level2Record index by binary_integer;
Type Level0Record is RECORD (Field1 Level1Collection);
Procedure NestedTypesProc (Param1 Level0Record);
```

In previous Oracle Database releases, the top-level type of the `NestedTypeProc` procedure, `parameter Param1, Level0Record`, is returned, and also an expanded description of all the nested types within `Level0Record`. For example:

```
SQL> select argument_name,type_subname,position,sequence,data_level
from user_arguments where object_name='NESTEDTYPESPROC';
ARGUMENT_NAME   TYPE_SUBNAME      POSITION   SEQUENCE  DATA_LEVEL
--------------- ----------------- ---------- ---------- ---------
PARAM1          LEVEL0RECORD             1          1           0
FIELD1          LEVEL1COLLECTION         1          2           1
                LEVEL2RECORD             1          3           2
FIELD1                                   1          4           3
```

In contrast, the same query in an 18.1 database returns the following:

```
ARGUMENT_NAME   TYPE_SUBNAME      POSITION   SEQUENCE  DATA_LEVEL
--------------- ----------------- ---------- ---------- ---------
PARAM1          LEVEL0RECORD             1          1           0
```

In releases earlier than Oracle Database 12c (12.1), PL/SQL package type descriptive metadata was not accessible in the way that metadata is accessible for top-level object types. With Top-level object types and collections, you can query `ALL_TYPES` and the associated user views, `ALL_TYPE_ATTRS`, and `ALL_COLL_TYPES`, to obtain type metadata. However, before Oracle Database 12.1, there was no way to obtain type metadata for PL/SQL package types, such as records and packaged collections. Function or procedure parameters that referenced those PL/SQL package types resulted in publishing all metadata about these types in the ARGUMENTS views, including any nested types.

The problem with this approach is that deeply nested types can consume extensive memory in the SYS tablespace. Also, because there is no way to share the type metadata in the ARGUMENTS views, each parameter with deeply nested types required its own redundant copy of the type metadata. The amount of metadata in the `ARGUMENTS` views and `SYS` tablespace, can lead to various issues, including PL/SQL compiler performance degradation. The degradation is caused because of the time it takes PL/SQL to update rows in the underlying dictionary tables.

In the Oracle Database 12.1 release, PL/SQL introduced enhanced support for package types, including the new user views, `ALL_PLSQL_TYPES`, `ALL_PLSQL_TYPE_ATTRS`, and `ALL_PLSQL_COLL_TYPES`. As the names imply, these views are similar to the `ALL_TYPES` view family. However, you can use the enhanced PL/SQL type views to query metadata about PL/SQL package types, instead of top-level object and collection types.

Because of the package types added with Oracle Database 12.1, there is no longer a need to insert large amounts of descriptive metadata into the ARGUMENTS views.

A single row of metadata that includes the type name is all that is required in the ARGUMENTS views for each parameter type. You can obtain a full description of the type name in a query against the PL/SQL type views, and any nested types.

`OCIDescribeAny()` is based on the same metadata used by the `ARGUMENTS` views. `OCIDescribeAny()` also returns a single row for each parameter type, instead of the multiple rows commonly returned before the change in Oracle Database 12.1.

`ALL/DBA/USER_ARGUMENTS` contains a new column type, `TYPE_OBJECT_TYPE`. To determine the type of the type described by `TYPE_OWNER`, `TYPE_NAME` and `TYPE_SUBNAME`, you use the `TYPE_OBJECT_TYPE` column. The possible values include `TABLE`, `VIEW`, `PACKAGE`, and `TYPE`.

If you prefer to continue to collect the `ALL_TYPES` and the associated user views, `ALL_TYPE_ATTRS` and ALL_COLL_TYPES in ARGUMENTS views, then you can set events to `events='10946, level 65536'`. Setting this event reverts the `ARGUMENTS` views back to the behavior in Oracle Database releases earlier than 12.1, in which DATA_LEVEL can be greater than `0`, and descriptive metadata for the type and any nested types is included in the view. If you make this change, then you must recompile affected packages after you set the event. When you recompile the affected packages, the compiler recollects the additional metadata. This event also reverts `OCIDescribeAny()` to the behavior in Oracle Database releases earlier than 12.1.

Starting in Oracle Database 12*c* release 1 (12.1.0.2), if you enter a procedure with no arguments, then the `ARGUMENTS` views do not have any rows. This change is an additional change that is separate from the row reduction change to `ARGUMENTS` views. Before Oracle Database 12.1.0.2, a procedure with no arguments was presented as a single row in the `ARGUMENTS` views.

USER/ALL/DBA_IDENTIFIERS User View Changes

Starting with Oracle Database 18c, PL/Scope is enhanced to capture additional information about user identifiers in PL/SQL code. The additional information includes constraints placed on the identifiers, and an indicator that notes when a function is a SQL builtin in PL/SQL.

The following columns are new in the USER/ALL/DBA_IDENTIFIERS views in Oracle Database 18c:

- `CHARACTER_SET`: This column contains the value of the character set clause, when the column is used in a variable identifier declaration. The possible values are `CHAR_CS`, `NCHAR_CS`, and `IDENTIFIER`, when the character set is derived from another variable identifier.
- `ATTRIBUTE`: This column contains the attribute value when `%`*`attribute`* is used in a variable declaration. The possible values are `ROWTYPE`, `TYPE`, and `CHARSET`.
- `CHAR_USED`: This column contains the type of the length constraint when a constraint is used in a string length constraint declaration. The possible values are `CHAR` and `BYTE`.
- `LENGTH`: This column contains the numeric length constraint value for a string length constraint declaration.
- `PRECISION`: This column contains the numeric precision when it is used in a variable declaration.
- `PRECISION2`: This column contains the numeric second precision value (for instance, interval types) used in a variable declaration.

- SCALE: This column contains the numeric scale value used in a variable declaration.
- LOWER_RANGE: This column contains the numeric lower range value used by a variable declaration with a range constraint.
- UPPER_RANGE: This column contains the numeric upper range value used by a variable declaration with a range constraint.
- NULL_CONSTRAINT: When a NULL constraint is used by a variable declaration, this column is set. The possible values are NULL, or NOT NULL.
- SQL_BUILTIN: When an identifier is a SQL builtin used in a SQL statement issued from PL/SQL, this column is set to YES. If the identifier is not a SQL builtin, then the column is set to NO.

PL/SQL EXTERNAL LIBRARY Changes

Starting with Oracle Database 18c, the methods change for how to create LIBRARY objects in an Oracle Database 18c PDB with a pre-defined PATH_PREFIX.

- When you create a new LIBRARY object in a PDB that has a predefined PATH_PREFIX, the LIBRARY must use a DIRECTORY object. The DIRECTORY object enforces the rules of PATH_PREFIX for the LIBRARY object. Failure to use a DIRECTORY object in the LIBRARY object results in a PLS-1919 compile-time error.
- If a database is plugged into a CDB as a PDB with a predefined PATH_PREFIX, then attempts to use a LIBRARY object that does not use a DIRECTORY object result in an ORA-65394 runtime error. The LIBRARY object is not invalidated. However, to make the LIBRARY useful (as opposed to always issuing a runtime error), you must recreate the LIBRARY object so that it uses a DIRECTORY object.

These changes enhance the security and manageability of LIBRARY objects in a PDB by accounting for the value of the PATH_PREFIX, which describes where the LIBRARY dynamic link library (DLL) can appear in the file system. The use of a DIRECTORY object also allows administrators to determine which users can access the DLL directory.

Symbolic Links and UTL_FILE

You cannot use UTL_FILE with symbolic links. Use directory objects instead.

The UTL_FILE_DIR symbolic link path is desupported in Oracle Database 18c and later releases. After an upgrade if applications address the database using symbolic links through UTL_FILE, then these links fail. Oracle recommends that you use directory objects. If necessary, you can create real files that are the targets of file names in UTL_FILE.

This desupport can affect any feature from an earlier release using symbolic links, including (but not restricted to) Oracle Data Pump, BFILEs, and External Tables. If you attempt to use an affected feature after upgrade, where that feature used symbolic links, you encounter ORA-29283: invalid file operation: path traverses a symlink. Before upgrade, to help you to identify symbolic link that you need to correct, use the Preupgrade Information Tool (preupgrade.jar). Oracle recommends that you instead use directory objects in place of symbolic links.

Example 10-1 Example of Error Messages with UTL_FILE And Symbolic Links

Applications that use symbolic links that address UTL_FILE encounter an error. For example. suppose you attempt to create a symbolic link, where `Ia.c` is a symbolic link file:

```
create or replace directory TEMP as '/home/PLSQL/TEMP';

declare
f utl_file.file_type;
begin
f := utl_file.fopen('TEMP','la.c','r');
end;
/
```

This command fails with the following errors:

```
ERROR at line 1:
ORA-29283: invalid file operation
ORA-06512: at "SYS.UTL_FILE", line 536
ORA-29283: invalid file operation
ORA-06512: at line 4
```

Deprecation of Direct Registration of Listeners with DBCA

Using Database Configuration Assistant (DBCA) to register Oracle Database to Oracle Internet Directory (OID) is deprecated in Oracle Database 18c.

Instead of using DBCA to migrate or register listeners to a database home during an upgrade, use Net Configuration Assistant or Net Manager to create a LISTENER.ORA file for the new release Oracle home, and then start this listener. You can also use DBCA to de-register and register listeners again to OID.

UNIFORM_LOG_TIMESTAMP_FORMAT Changes in INIT.ORA

By default, the format of timestamps is different in Oracle Database 12*c* release 2 (12.2) and later releases. To view alert logs, use the Oracle Database utility Automatic Diagnostic Repository Command Interpreter (ADRCI) utility.

If you use scripts to parse the alert log for timestamp dates, then be aware that the default value for timestamp formats is set by the `init.ora` parameter UNIFORM_LOG_TIMESTAMP_FORMAT. The default value for this parameter is TRUE. When TRUE, the timestamp format changes from a day-month-year-time format to a year-month-day-time format. For example: `2017-05-17T10:00:54.799968+00:00`.

You can change to the timestamp format used in previous releases by changing the value of UNIFORM_LOG_TIMESTAMP_FORMAT to FALSE. You can also use scripts to parse `log.xml` instead of the alert log.

Oracle provides a dedicated command-line utility to find and analyze Oracle errors and tracefiles, called Automatic Diagnostic Repository Command Interpreter (ADRCI) Oracle recommends that you use the ADRCI utility for error management.

For example, you can use the ADRCI command `show alert` to view the alert log:

```
$ oracle@user> adrci
adrci> show alert -tail -f
```

ADRCI also enables you to use the `show log` command to pass predicates for a query. For example:

```
adrci> show log -p "message_text like '%tablespace%'"
```

Refer to *Oracle Database Utilities* for more information about how to use the ADRCI utility.

Related Topics

- *Oracle Database Utilities*

Deprecated Features in Oracle Database 18c

Review the deprecated features listed in this section to prepare to use alternatives after you upgrade.

> **Note:**
>
> The non-CDB architecture was deprecated in Oracle Database 12c. It can be desupported and unavailable in a release after Oracle Database 19c .

- Data Guard MAX_CONNECTIONS Attribute is Deprecated
  The `MAX_CONNECTIONS` attribute of the `LOG_ARCHIVE_DEST_`*n* parameter for Data Guard redo transport is deprecated in Oracle Database 18c.
- Extended Datatype Support (EDS) is Deprecated
  Extended Datatype Support (EDS) is deprecated in Oracle Database 18c .
- GET_* Functions Deprecated in the DBMS_DATA_MINING Package
  Starting in Oracle Database 18c, the `GET_*` functions in `DBMS_DATA_MINING` are deprecated. Use the Oracle Data Mining (ODM) Model Details views instead.
- Package DBMS_XMLQUERY is deprecated
  The PL/SQL package `DBMS_XMLQUERY` is deprecated in Oracle Database 18c. Use `DBMS_XMLGEN` instead.
- Package DBMS_XMLSAVE is Deprecated
  The PL/SQL package `DBMS_XMLSAVE` is deprecated in Oracle Database 18c. Use `DBMS_XMLSTORE` instead.
- Deprecated Columns in Oracle Label Security Views
  Starting in Oracle Database 18c, The LABELS column is deprecated in the ALL_SA_USER_LABELS and DBA_SA_USER_LABELS views.
- Returning JSON True or False Values using NUMBER is Deprecated
  Starting with Oracle Database 18c , the option to specify a SQL NUMBER value (1 or 0) as the return value of a JSON value of true or false is deprecated.

- Deprecation of MAIL_FILTER in Oracle Text
  Starting with Oracle Database 18c, the use of `MAIL_FILTER` in Oracle Text is deprecated. Before adding email to the database, filter e-mails to indexable plain text, or to HTML.
- Deprecation of asmcmd showversion Option
  Starting with Oracle Database 18c, the command options for `asmcmd showversion` are replaced with new `asmcmd` options.
- Deprecation of NEWS_SECTION_GROUP in Oracle Text
  Starting with Oracle Database 18c, use of `NEWS_SECTION_GROUP` is deprecated in Oracle Text. Use external processing instead.
- Oracle Net Services Support for SDP is Deprecated
  Starting with Oracle Database 18c, the Oracle Net Services support for Sockets Direct Protocol (SDP) is deprecated.
- Deprecation of Flex Cluster (Hub/Leaf) Architecture
  Starting with Oracle Database 18c, Leaf nodes are deprecated as part of Oracle Flex Cluster architecture.
- Deprecation of PRODUCT_USER_PROFILE Table
  Starting in Oracle Database 18c, the SQL*Plus table `PRODUCT_USER_PROFILE` (PUP) table is deprecated.

Data Guard MAX_CONNECTIONS Attribute is Deprecated

The `MAX_CONNECTIONS` attribute of the `LOG_ARCHIVE_DEST_`*n* parameter for Data Guard redo transport is deprecated in Oracle Database 18c.

Oracle Database 11*g* Release 1 (11.1) introduced the new streaming asynchronous model for redo transport. Using the `MAX_CONNECTIONS` attribute setting no longer provides any benefit when Oracle Data Guard is resolving gaps in the archive log files.

Extended Datatype Support (EDS) is Deprecated

Extended Datatype Support (EDS) is deprecated in Oracle Database 18c .

The Extended Datatype Support (EDS) feature provides a mechanism for logical standbys to support certain Oracle data types that lack native redo-based support. For example, EDS was used to replicate tables with SDO_GEOMETRY column. However, starting with Oracle Database 12c Release 2 (12.2), there are no EDS-supported Oracle data types that are not supported natively by Logical data or GoldenGate. This feature is now obsolete.

GET_* Functions Deprecated in the DBMS_DATA_MINING Package

Starting in Oracle Database 18c, the `GET_*` functions in `DBMS_DATA_MINING` are deprecated. Use the Oracle Data Mining (ODM) Model Details views instead.

In Oracle Database 12c Release 1, and earlier releases, the `DBMS_DATA_MINING` package supports a separate `GET_MODEL_DETAILS` function for each data mining algorithm. Global details are also available for Generalized Linear Models, Expectation Maximization, Singular Value Decomposition, and Association Rules. There are many `DBMS_DATA_MINING Get_*` functions. For example:

- `GET_MODEL_DETAILS`

- DBMS_DATA_MINING.GET_MODEL_TRANSFORMATIONS

For example, the Model detail view for Decision Tree describes the split information view, node statistics view, node description view, and the cost matrix view.

Starting with Oracle Database 18c, Oracle recommends that you replace the GET_MODEL_DETAILS_XML functions with the Oracle Data Mining Model Details views. The split information view DM$VP*model_name* describes the decision tree hierarchy, in which you append the name of the Oracle Data Mining model to the view prefix.

Package DBMS_XMLQUERY is deprecated

The PL/SQL package DBMS_XMLQUERY is deprecated in Oracle Database 18c. Use DBMS_XMLGEN instead.

DBMS_XMLQUERY provides database-to-XMLType functionality. Oracle recommends that you replace calls to DBMS_XMLQUERY with DBMS_XMLGEN. DBMS_XMLGEN is written in C, and compiled into the kernel, so it provides higher performance.

Package DBMS_XMLSAVE is Deprecated

The PL/SQL package DBMS_XMLSAVE is deprecated in Oracle Database 18c. Use DBMS_XMLSTORE instead.

The DBMS_XMLSAVE package is part of the Oracle XML SQL Utility. It is used to insert, update, and delete data from XML documents in object-relational tables. Oracle recommends that you replace DBMS_XMLSAVE calls with DBMS_XMLSTORE. DBMS_XMLSTORE is written in C, and compiled into the kernel, so it provides higher performance.

For example: to replace DBMS_XMLSAVE, you can create a wrapper function or procedure that you used to call DBMS_XMLSAVE on an earlier release Oracle Database, and change the call to DBMS_XMLSTORE. Or you can create a synonym:

For example: to replace DBMS_XMLSAVE, you can create a wrapper function or procedure that you used to call DBMS_XMLSAVE on an earlier release Oracle Database, and change the call to DBMS_XMLSTORE. Or you can create a synonym:

```
CREATE OR REPLACE PUBLIC SYNONYM DBMS_XMLSAVE FOR DBMS_XMLSTORE;
GRANT EXECUTE ON DBMS_XMLSAVE TO PUBLIC;
```

Deprecated Columns in Oracle Label Security Views

Starting in Oracle Database 18c, The LABELS column is deprecated in the ALL_SA_USER_LABELS and DBA_SA_USER_LABELS views.

Table 10-2 Deprecated columns in Oracle Label Security Views

Data Dictionary View	Deprecated Column
ALL_SA_USER_LABELS	LABELS
ALL_SA_USERS	USER_LABELS
DBA_SA_USER_LABELS	LABELS
DBA_SA_USERS	USER_LABELS

The information in the LABELS and USER_LABELS columns is redundant. This information is displayed in other columns in these data dictionary views.

Returning JSON True or False Values using NUMBER is Deprecated

Starting with Oracle Database 18c , the option to specify a SQL NUMBER value (1 or 0) as the return value of a JSON value of true or false is deprecated.

Oracle Database 12*c* release 1 (12.1) provided support for JSON data, including the function of specifying NUMBER as the type of a column that is returned. The option to specify NUMBER is deprecated. Instead of specifying NUMBER as the output for JSON data for true/false queries, you can use the default SQL value returned for a JSON Boolean value, and specify the string as 'true' or 'false'. If you have an application that requires a numeric value, then you can return the Boolean JSON value as a SQL VARCHAR2 value, and then test that value and return a SQL NUMBER value as the result of that test.

Deprecation of MAIL_FILTER in Oracle Text

Starting with Oracle Database 18c, the use of MAIL_FILTER in Oracle Text is deprecated. Before adding email to the database, filter e-mails to indexable plain text, or to HTML.

MAIL_FILTER is based on an obsolete email protocol, RFC-822. Modern email systems do not support RFC-822. There is no replacement.

Deprecation of asmcmd showversion Option

Starting with Oracle Database 18c, the command options for `asmcmd showversion` are replaced with new `asmcmd` options.

In place of the command `asmcmd showversion --softwarepatch`, use the new option `asmcmd showpatches -l`. In place of the command `asmcmd showversion --releasepatch`, use the new option `asmcmd showversion --active`.

Deprecation of NEWS_SECTION_GROUP in Oracle Text

Starting with Oracle Database 18c, use of NEWS_SECTION_GROUP is deprecated in Oracle Text. Use external processing instead.

If you want to index USENET posts, then preprocess the posts to use BASIC_SECTION_GROUP or HTML_SECTION_GROUP within Oracle Text. USENET is rarely used commercially.

USENET currently is rarely used for serious purpose. Performing index processing using this section group type is obsolete.

Oracle Net Services Support for SDP is Deprecated

Starting with Oracle Database 18c, the Oracle Net Services support for Sockets Direct Protocol (SDP) is deprecated.

Oracle recommends that you use TCP as an alternative.

Deprecation of Flex Cluster (Hub/Leaf) Architecture

Starting with Oracle Database 18c, Leaf nodes are deprecated as part of Oracle Flex Cluster architecture.

With continuous improvements in the Oracle Clusterware stack towards providing shorter reconfiguration times in case of a failure, Leaf nodes are no longer necessary for implementing clusters that meet customer needs, either for on-premises, or in the Cloud.

Deprecation of PRODUCT_USER_PROFILE Table

Starting in Oracle Database 18c, the SQL*Plus table `PRODUCT_USER_PROFILE` (PUP) table is deprecated.

The only use for the `PRODUCT_USER_PROFILE` (PUP) table is to provide a mechanism to control product-level security for SQL*Plus. Starting with Oracle Database 18c, this mechanism is no longer relevant. This SQL*Plus product-level security feature will be unavailable in Oracle Database 19c. Oracle recommends that you protect data by using Oracle Database settings, so that you ensure consistent security across all client applications.

Desupported Features in Oracle Database 18c

Review this list of desupported features as part of your upgrade planning.

- Oracle Administration Assistant for Windows is Desupported
  The Oracle Administration Assistant tool for Windows is desupported in Oracle Database 18c.
- Oracle Multimedia DICOM Desupported Features
  Several Oracle Multimedia DICOM features are desupported in Oracle Database 18c. Replace DICOM with Oracle SecureFiles and third-party DICOM products.
- Oracle Multimedia Java Client Classes Desupported
  Oracle Multimedia proxy classes and Oracle Multimedia servlet and JSP classes are desupported.
- Oracle XML DB Desupported Features
  Starting withOracle Database 18c, schema subprograms in `DBMS_XMLSCHEMA`, many `DBMS_XDB` subprograms, and many other Oracle XML DB schema features are desupported.
- ODP.NET, Managed Driver - Distributed Transaction DLL Desupported
  Oracle is desupporting the `Oracle.ManagedDataAccessDTC.dll` file in Oracle Database 18c.
- Data Guard Broker DGMGRL ALTER Syntax is Desupported
  Starting with Oracle Database 18c, the Oracle Data Guard broker ALTER command in DGMGRL is desupported.
- Desupport of CRSUSER on Microsoft Windows Systems
  The `crsuser` utility and the `CRSToken` method to change the Windows service user is desupported in Oracle Database 18c.

Oracle Administration Assistant for Windows is Desupported

The Oracle Administration Assistant tool for Windows is desupported in Oracle Database 18c.

Oracle Administration Assistant for Windows is desupported in the current database release. Oracle Administration Assistant for Windows was a tool for creating database administrators, operators, users, and roles in Windows. Oracle Administration Assistant also enabled database services, startup and shutdown configurations, and Windows Registry parameter management. There is no replacement.

Oracle Multimedia DICOM Desupported Features

Several Oracle Multimedia DICOM features are desupported in Oracle Database 18c. Replace DICOM with Oracle SecureFiles and third-party DICOM products.

Digital Imaging and Communications in Medicine (DICOM) is a medical imaging technology that supports the connectivity of radiological devices. Oracle's native DICOM feature is deprecated, and parts of it are desupported in this release. The desupport of Oracle Multimedia DICOM includes the following features:

- Oracle Multimedia DICOM protocol
- Oracle Multimedia DICOM mid-tier support
- Oracle Multimedia Oracle DICOM Component for WebCenter integration (DICOM/UCM)

The following Oracle Multimedia DICOM features continue to be deprecated:

- DICOM support in Oracle Multimedia ORDImage object
- Oracle Multimedia DICOM objects and packages

There is no replacement for Oracle Multimedia DICOM. Oracle recommends that you replace Oracle Multimedia DICOM by using Oracle SecureFiles with third-party products for DICOM functionality. For example: Use third-party DICOM features to carry out metadata management, DICOM image conversion, and so on.

Oracle Multimedia Java Client Classes Desupported

Oracle Multimedia proxy classes and Oracle Multimedia servlet and JSP classes are desupported.

Oracle Multimedia Java client is desupported in Oracle Database 18c for the following classes:

- Oracle Multimedia proxy classes, including DICOM proxy classes
- Oracle Multimedia servlet/jsp classes

To develop Java applications that manage multimedia content within Oracle Databases, Oracle recommends that you embed PL/SQL blocks in Java.

Oracle XML DB Desupported Features

Starting withOracle Database 18c, schema subprograms in DBMS_XMLSCHEMA, many DBMS_XDB subprograms, and many other Oracle XML DB schema features are desupported.

In Oracle Database 12*c* release 1 (12.1), the PL/SQL package DBMS_XDB_CONFIG was introduced. At the same time, all Oracle XML DB configuration functions, procedures, and constants that were moved from package DBMS_XDB to DBMS_XDB_CONFIG. were deprecated, and a series of other DBMS_XMLSCHEMA, DBMS_XDB subprograms, and other schema features were deprecated. These components are now desupported.

Desupported PL/SQL subprograms in package DBMS_XMLSCHEMA

The following PL/SQL subprograms in package DBMS_XMLSCHEMA are desupported:

- generateSchema
- generateSchemas

There are no replacements for these constructs. There is no workaround for this change.

Desupported Oracle XML DB Configuration Functions, Procedures, and Constants

All Oracle XML DB configuration functions, procedures, and constants that were moved from package DBMS_XDB to DBMS_XDB_CONFIG are desupported. Use DBMS_XDB_CONFIG.

The following list of subprograms are desupported in package DBMS_XDB:

- ADDHTTPEXPIREMAPPING
- ADDMIMEMAPPING
- ADDSCHEMALOCMAPPING
- ADDSERVLET
- ADDSERVLETMAPPING
- ADDSERVLETSECROLE
- ADDXMLEXTENSION
- CFG_GET
- CFG_REFRESH
- CFG_UPDATE
- DELETEHTTPEXPIREMAPPING
- DELETEMIMEMAPPING
- DELETESCHEMALOCMAPPING
- DELETESERVLET
- DELETESERVLETMAPPING
- DELETESERVLETSECROLE

- DELETEXMLEXTENSION
- GETFTPPORT
- GETHTTPPORT
- GETLISTENERENDPOINT
- SETFTPPORT
- SETHTTPPORT
- SETLISTENERENDPOINT
- SETLISTENERLOCALACCESS

The following constants are desupported in package DBMS_XDB:

- XDB_ENDPOINT_HTTP
- XDB_ENDPOINT_HTTP2
- XDB_PROTOCOL_TCP
- XDB_PROTOCOL_TCPS

Desupported Oracle XQuery Functions

The following Oracle XQuery functions are desupported. Use the corresponding standard XQuery functions instead. Corresponding functions are the functions that have the same names, but that use the namespace prefix fn.

- ora:matches . Use fn:matches instead
- ora:replace. Use fn:replace instead

ODP.NET, Managed Driver - Distributed Transaction DLL Desupported

Oracle is desupporting the Oracle.ManagedDataAccessDTC.dll file in Oracle Database 18c.

Oracle provided a native managed distributed transaction support for Oracle Data Provider for .NET (ODP.NET), Managed Driver using Oracle.ManagedDataAccessDTC.dll. In .NET Framework 4.5.2, Microsoft introduced its own native managed distributed transaction implementation, which managed ODP.NET used. The new .NET Framework made the Oracle.ManagedDataAccessDTC.dll unnecessary. Moreover, Microsoft has desupported all .NET Framework 4 versions earlier than 4.5.2. In accordance with Microsoft policy, Oracle is desupporting the Oracle.ManagedDataAccessDTC.dll file.

The desupport includes removing the UseManagedDTC .NET configuration file parameter, and Oracle.ManagedDataAccessDTC.dll.

Data Guard Broker DGMGRL ALTER Syntax is Desupported

Starting with Oracle Database 18c, the Oracle Data Guard broker ALTER command in DGMGRL is desupported.

The ALTER command syntax in the Data Guard broker DGMGRL command-line interface was deprecated in Oracle Database 10g Release 1 and replaced with the EDIT CONFIGURATION, EDIT DATABASE, and EDIT INSTANCE syntax.

Desupport of CRSUSER on Microsoft Windows Systems

The `crsuser` utility and the `CRSToken` method to change the Windows service user is desupported in Oracle Database 18c.

In Oracle Grid Infrastructure releases before Release 12c (12.1), it was supported to use the `crsuser` utility with Oracle Real Application Clusters (Oracle RAC) to modify the database logon properties of the Oracle Database service from `LocalSystem` to a user ID.

Oracle introduced the Oracle Home User system privileges role for the DB home in Oracle Grid Infrastructure 12c Release 1 (12.1). This role makes the need for the `crsuser` functionality unnecessary. The `crsuser` facility was also previously used to create user-defined CRS resources that ran as a Windows user other than `LocalSystem`. However, Oracle Grid Infrastructure 12*c* Release 1 (12.1) and later releases provide that same functionality with `crsctl add wallet -type OSUSER` The `crsuser` feature no longer works. It is no longer developed or supported.

For more information about the `crsctl add wallet -type OSUSER` command, refer to *Oracle Clusterware Administration and Deployment*.

Related Topics

- *Oracle Clusterware Administration and Deployment Guide*

Desupported Initialization Parameters in Oracle Database 18c

Review this list of desupported initialization parameters for changes and replacements in parameter settings in this release.

- Desupport of STANDBY_ARCHIVE_DEST Initialization Parameter
  Support for the initialization parameter `STANDBY_ARCHIVE_DEST` is removed in Oracle Database 18c.
- Desupport of UTL_FILE_DIR Initialization Parameter
  Starting in Oracle Database 18c, the `UTL_FILE_DIR` parameter is no longer supported. Instead, specify the name of a directory object.

Desupport of STANDBY_ARCHIVE_DEST Initialization Parameter

Support for the initialization parameter `STANDBY_ARCHIVE_DEST` is removed in Oracle Database 18c.

Oracle Database 11*g* Release 2 (11.2) included an increase to 31 of the parameters `LOCAL` and `REMOTE` archiving `LOG_ARCHIVE_DEST_`*n*. This increase, and the `ALTERNATE` attribute enhancements to provide high availability for local and remote archiving, provides you with more control over the results after an archiving destination fails. Because of these enhancements, `STANDBY_ARCHIVE_DEST` is not required or practical to use.

Desupport of UTL_FILE_DIR Initialization Parameter

Starting in Oracle Database 18c, the `UTL_FILE_DIR` parameter is no longer supported. Instead, specify the name of a directory object.

The UTL_FILE_DIR initialization parameter is no longer listed in V$SYSTEM_PARAMETER and related views. If you attempt to set this parameter, then the attempt fails. If you attempt to specify an operating system file directly by using the LOCATION parameter of UTL_FILE.FOPEN, or by using the LOCATION parameter of FOPEN_NCHAR, then those attempts also fail. Specify the name of a directory object instead.

The security model for the use of a directory object for UTL_FILE and other Oracle Database subsystems is secure, because there is a clear privilege model. However, the use of an explicit operating system directory is insecure, because there is no associated privilege model. The notice of deprecation for the UTL_FILE_DIR initialization parameter was given in Oracle Database 12c Release 2 (12.2). With Oracle Database 18c, the parameter is now desupported.

UTL_FILE Package Symbolic Link in Directory Paths Not Supported

Using the UTL_FILE package to access a symbolic link fails in the new Oracle Database release. To avoid the issue, you must change the directory object and the file name, so that neither contains a symbolic link. This desupport can affect any feature from an earlier release using symbolic links, including (but not restricted to) Oracle Data Pump, BFILEs, and External Tables. If you attempt to use an affected feature after upgrade, where that feature used symbolic links, you encounter `ORA-29283: invalid file operation: path traverses a symlink`.

Deprecation of Oracle Multimedia

Starting in Oracle Database 18c, Oracle Multimedia is deprecated. Oracle Multimedia will be desupported in Oracle Database 19c.

Oracle recommends that you store multimedia content in SecureFiles LOBs, and use open source or third-party products such as Piction for image processing and conversion. Oracle Locator is not affected by the deprecation of Oracle Multimedia.

Terminal Release of Oracle Streams

Oracle Database 18c is the terminal release for Oracle Streams support. Oracle Streams will be desupported from Oracle Database 19c onwards.

Oracle Streams was deprecated in Oracle Database 12c Release 1 (12.1). It does not support features introduced in Oracle Database 12c and later releases, including the multitenant architecture, the LONG VARCHAR data type, long identifiers, and other features. Oracle GoldenGate is the replication solution for Oracle Database.

Behavior Changes, Deprecations and Desupports in Oracle Database 12c Release 2 (12.2)

Review for descriptions of Oracle Database 12c Release 2 (12.2) changes.

- Behavior Changes in Oracle Database 12c Release 2 (12.2)
  Review these behavior changes to help plan for upgrades to Oracle Database 12c release 2 (12.2)

- Deprecated Initialization Parameters in Oracle Database 12c Release 2 (12.2)
  To understand changes and replacements in parameter settings, review the parameters deprecated in the 12.2 release. These parameters can be removed in a later release.
- Deprecated Features in Oracle Database 12c Release 2 (12.2)
  Review the deprecated features listed in this section to prepare to use alternatives after you upgrade.
- Desupported Initialization Parameters in Oracle Database 12c Release 2 (12.2)
  Review this list of desupported initialization parameters for changes and replacements in parameter settings in this release.
- Desupported Features in Oracle Database 12c Release 2 (12.2)
  Review this list of desupported features as part of your upgrade planning.

Behavior Changes in Oracle Database 12c Release 2 (12.2)

Review these behavior changes to help plan for upgrades to Oracle Database 12c release 2 (12.2)

- Initialization Parameter Default Changes in Oracle Database 12c Release 2 (12.2)
  Review this list of initialization parameter default setting changes for Oracle Database 12c release 2 (12.2).
- Database Upgrade Assistant (DBUA) Enhancements and Changes
  Oracle Database 12*c* release 2 (12.2) includes several enhancements to DBUA, and some features have been removed or modified.
- Enhancements to Oracle Data Guard Broker and Rolling Upgrades
  Starting with Oracle Database 12*c* release 2 (12.2), Oracle Data Guard Broker has more features to assist rolling upgrades.
- About Changes in Default SGA Permissions for Oracle Database
  Starting with Oracle Database 12c Release 2 (12.2.0.1), by default, permissions to read and write to the System Global Area (SGA) are limited to the Oracle software installation owner.
- Network Access Control Lists and Upgrade to Oracle Database 12c
  Network access control lists (ACLs) are implemented as Real Application Security ACLs in 12c, and existing ACLs are migrated from XML DB ACLs and renamed during upgrade.
- Parallel Upgrade Utility Batch Scripts
  In Oracle Database 12*c* Release 2 and later releases, you can run the Parallel Upgrade Utility using command-line batch scripts.`catupgrd.sql` is no longer distributed.
- Unified Auditing AUDIT_ADMIN and AUDIT_VIEWER Roles Changes
  You can find it necessary to rename or drop `AUDIT_ADMIN` and `AUDIT_VIEWER` roles before upgrading.
- Oracle Update Batching Batch Size Settings Disabled
  Oracle update batching settings are disabled in Oracle Database 12c release 2 (12.2). Use JDBC batching instead.
- About Upgrading Tables Dependent on Oracle-Maintained Types
  Starting with Oracle Database 12*c* release 2 (12.2), you can run the Parallel Upgrade Utility with the `-T` option to set tables to READ ONLY.

- Case-Insensitive Passwords and ORA-1017 Invalid Username or Password
  The Oracle Database 12*c* release 2 (12.2) default authentication protocol is 12 (Exclusive Mode). This protocol requires case-sensitive passwords for authentication. Review your options if you have earlier release password versions.
- About Deploying Oracle Grid Infrastructure Using Oracle Fleet Patching and Provisioning
  Oracle Fleet Patching and Provisioning (Oracle FPP) is a software lifecycle management method for provisioning and maintaining Oracle homes. Oracle Fleet Patching and Provisioning enables mass deployment and maintenance of standard operating environments for databases, clusters, and user-defined software types.
- Restrictions Using Zero Data Loss Recovery Appliance Release 12.1 Backups
  Zero Data Loss Recovery Appliance release 12.1 does not support backups from protected database clients using Oracle Database 12*c* release 2 (12.2).
- Client and Foreground Server Process Memory Changes
  To increase optimization and performance, the Oracle Database Client and server process memory requirements are greater than in previous releases.

Initialization Parameter Default Changes in Oracle Database 12c Release 2 (12.2)

Review this list of initialization parameter default setting changes for Oracle Database 12c release 2 (12.2).

OPTIMIZER_ADAPTIVE_PLANS and OPTIMIZER_ADAPTIVE_STATISTICS

OPTIMIZER_ADAPTIVE_FEATURE functions are replaced by two new parameters: OPTIMIZER_ADAPTIVE_PLANS, and OPTIMIZER_ADAPTIVE_STATISTICS.

OPTIMIZER_ADAPTIVE_PLANS controls adaptive plans. It is set by default to TRUE. When set to TRUE, this parameter determines alternate execution plans built with alternative choices that are based on statistics collected as a query executes.

OPTIMIZER_ADAPTIVE_STATISTICS controls adaptive statistics. It is set by default to FALSE. When set to TRUE, the optimizer augments the statistics gathered in the database with adaptive statistics gathered at SQL statement parse time to improve the quality of SQL execution plans. Some query shapes are too complex to rely upon base table statistics alone. The optimizer augments them with adaptive statistics to determine more accurately the best SQL execution plan.

SQL92_SECURITY Initialization Parameter Default is TRUE

The SQL standard specifies that security administrators should be able to require that users have SELECT privilege on a table when running an UPDATE or DELETE statement that references table column values in a WHERE or SET clause. SQL92_SECURITY specifies whether users must have been granted the SELECT object privilege to execute such UPDATE or DELETE statements.

Starting in Oracle Database 12c release 2 (12.2), the default setting for this parameter changes from FALSE to TRUE.

When this parameter is set to TRUE, users must have SELECT privilege on the object being deleted or updated.

Related Topics

- *Oracle Database Reference*

Database Upgrade Assistant (DBUA) Enhancements and Changes

Oracle Database 12*c* release 2 (12.2) includes several enhancements to DBUA, and some features have been removed or modified.

In response to customer requests, and to improve functionality, Database Upgrade Assistant (DBUA) includes new features and code enhancements. Also, some features in previous releases have been removed.

DBUA New Features

DBUA includes the following new features for Oracle Database 12*c* release 2 (12.2):

- Selective PDB Plug-In Upgrades: You can plug in a PDB created in a previous release into a release 12.2 multitenant architecture CDB environment, and upgrade the PDB using DBUA started from the release 12.2 CDB home

  You can unplug PDBs from a CDB, upgrade the CDB and any PDBs plugged in to the CDB, and then plug in earlier release PDBs and upgrade them using DBCA.
- Priority-Based PDB Upgrades: You can set priority for PDB upgrades, so that higher priority PDBs are upgraded first.
- Retry and Ignore Functionality: You can fix errors and retry upgrades, or select to ignore certain errors and continue upgrades.
- Pause and Continue Functionality: You can stop the upgrade, and continue the upgrade at a later time.
- Standalone Prerequisite Checks: You can run DBUA with the new `-executePrereqs` option to check prerequisites for upgrades at any time.
- Listener Configuration During Database Moves: You can configure the database with a new listener during a database move operation.
- Improved Logging Mechanism: DBUA now has time-stamped logs.
- Performance Enhancements: DBUA includes code enhancements that reduce the number of instance restarts during the upgrade process.
- Enhanced Error Reporting: All DBUA errors are reported using the error code prefix `DBT`, and all errors are reported as a list on a progress page, instead of being presented in message windows.

DBUA Removed Features

The following DBUA features available in previous releases are removed in Oracle Database 12*c* release 2 (12.2):

- Data Files Move: Data files can no longer be moved during upgrades.
- Database Renames During Upgrades: It is no longer supported to rename Oracle Database names during the upgrade.
- Degree of Parallelism Selection Removed from DBUA: The default parallelism is calculated depending on the use case.
  - Upgrade: The default parallelism using DBUA is the same value used by the Parallel Upgrade Utility for manual upgrades. However, in an upgrade

operation, you can override the default by specifying the number of cores that you want to use.

  - Recompile: The default parallelism for object recompilation is determined by the `utlrp` script used in manual upgrade.

- Recompile parallelism is the same value as the upgrade parallelism by default.
- Changing Diagnostic and Audit Dest No Longer Available: You can only change the Diagnostic and Audit destination by using the DBUA command-line option `-initParam`.
- Remote DBUA Desupported: In previous releases, DBUA had an option on Windows platforms for supporting Oracle Database remote upgrades. This feature is desupported.

Enhancements to Oracle Data Guard Broker and Rolling Upgrades

Starting with Oracle Database 12*c* release 2 (12.2), Oracle Data Guard Broker has more features to assist rolling upgrades.

Oracle Data Guard Broker now supports Oracle Active Data Guard rolling upgrade. Oracle Active Data Guard rolling upgrade was introduced in Oracle Database 12*c* release 1 (12.1). It simplifies the execution of the transient logical database rolling upgrade process by automating many manual steps in a simple PL/SQL package (DBMS_ROLLING). In addition to making database rolling upgrades simpler, the automated process is much more reliable. Oracle Data Guard broker can now direct Oracle Active Data Guard rolling upgrades from the DGMGRL command-line interface. Broker support also adds substantial simplification to the rolling upgrade process by transparently handling redo transport destination settings and other tasks.

In Oracle Database 12*c* release 2 (12.2) and later releases, when you perform a rolling upgrade using the DBMS_ROLLING PL/SQL package, you no longer have to disable the broker. In addition, the broker now reports when a rolling upgrade is in place, and tracks its status. The status information is displayed in the output of the DGMGRL commands SHOW CONFIGURATION and SHOW DATABASE.

Using Oracle Data Guard Broker to manage database rolling upgrades can simplify the upgrade process by minimizing your downtime and risk when introducing change to production environments.

Related Topics

- *Oracle Data Guard Broker*

About Changes in Default SGA Permissions for Oracle Database

Starting with Oracle Database 12c Release 2 (12.2.0.1), by default, permissions to read and write to the System Global Area (SGA) are limited to the Oracle software installation owner.

In previous releases, both the Oracle installation owner account and members of the OSDBA group had access to shared memory. The change in Oracle Database 12c Release 2 (12.2) and later releases to restrict access by default to the Oracle installation owner account provides greater security than previous configurations. However, this change may prevent DBAs who do not have access to the Oracle installation owner account from administering the database.

The Oracle Database initialization parameter ALLOW_GROUP_ACCESS_TO_SGA determines if the Oracle Database installation owner account (oracle in Oracle documentation examples) is the only user that can read and write to the database System Global Area (SGA), or if members of the OSDBA group can read the SGA. In Oracle Database 12c Release 2 (12.2) and later releases, the default value for this parameter is FALSE, so that only the Oracle Database installation owner has read and write permissions to the SGA. Group access to the SGA is removed by default. This change affects all Linux and UNIX platforms.

If members of the OSDBA group require read access to the SGA, then you can change the initialization parameter ALLOW_GROUP_ACCESS_TO_SGA setting from FALSE to TRUE. Oracle strongly recommends that you accept the default permissions that limit access to the SGA to the oracle user account.

Related Topics

- *Oracle Database Reference*

Network Access Control Lists and Upgrade to Oracle Database 12c

Network access control lists (ACLs) are implemented as Real Application Security ACLs in 12c, and existing ACLs are migrated from XML DB ACLs and renamed during upgrade.

During Oracle Database upgrades to 12c Release 1 (12.1) and later releases, network access control in Oracle Database is implemented using Real Application Security access control lists (ACLs). Existing ACLs in XDB are migrated during upgrade. Existing APIs in the DBMS_NETWORK_ACL_ADMIN PL/SQL package and catalog views are deprecated. These deprecated views are replaced with new equivalents in Oracle Database 12c.

Starting with Oracle Database 12c Release 1 (12.1), you can grant network privileges by appending an access control entry (ACE) to a host ACL using DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE. If you append an ACE to a host that has no existing host ACL, then a new host ACL is created implicitly. If the host ACL exists, then the ACEs are appended to the existing ACL.

How Changing to Real Application Security ACLS Affects You

During upgrades, the following changes are made:

- Existing network ACLs are migrated from Oracle Database 11g XML DB to Oracle Database 12c Real Application Security. All privileges of the existing ACLs are preserved during this migration.
- Existing ACLs are renamed.

What You Need To Do Before Upgrades

- Check for existing Network ACLs before the upgrade.
- Preserve existing network ACLs and privileges (DBA_NETWORK_ACLS and DBA_NETWORK_ACL_PRIVILEGES) in an intermediate staging table. Preserving the existing privileges in a table enables you to restore them if the automatic migration fails, or if you want to roll back an upgrade.

Related Topics

- *Oracle Database Security Guide*

- *Oracle Database Reference*

Parallel Upgrade Utility Batch Scripts

In Oracle Database 12*c* Release 2 and later releases, you can run the Parallel Upgrade Utility using command-line batch scripts.`catupgrd.sql` is no longer distributed.

In Oracle Database 12*c* Release 2 (12.2) and later releases, you can run the Parallel Upgrade Utility (`catctl.pl`) from the command line by entering the shell commands `dbupgrade` for Linux and Unix, and `dbupgrade.com` for Microsoft Windows. These shell scripts call the `catctl.pl` script from the upgrade binary home. You can either run these scripts with default values, or you can run them with the same input parameters that you use to run `catctl.pl` from the Perl prompt.

Related Topics

- About the Parallel Upgrade Utility for Oracle Database (CATCTL.PL and DBUPGRADE)

Unified Auditing AUDIT_ADMIN and AUDIT_VIEWER Roles Changes

You can find it necessary to rename or drop `AUDIT_ADMIN` and `AUDIT_VIEWER` roles before upgrading.

In Oracle Database 12*c*, if you use Unified Auditing, then you can have two AUDSYS roles in your Oracle Database 11*g* release 2 (11.2.0.4) and earlier releases that affect upgrading: `AUDIT_ADMIN` and `AUDIT_VIEWER`. Because of changes in these roles, you must drop these earlier release users or user roles before you can upgrade to Oracle Database 12*c* release 1 (12.1) or later.

If you have created `AUDIT_ADMIN` and `AUDIT_VIEWER` users or roles with Oracle Database 12*c* release 1 (12.1), then you do not need to drop these users or roles.

Only drop the `AUDSYS` schema and the `AUDIT_ADMIN` and `AUDIT_VIEWER` roles if both of the following conditions are true:

- The version from which you are upgrading is earlier than Oracle Database 12*c* release 1 (12.1)
- You have created a custom schema with the name `AUDSYS`

If you are affected by this requirement, and you cannot drop these `AUDSYS` roles, then select the `UNIFIED_AUDIT_TRAIL` view, create your own table, using similar definitions, and use this table to take a backup of the Unified Audit data. Oracle recommends that you also perform this procedure if you want to preserve your ability to downgrade to your earlier release database.

The Pre-Upgrade Information Tool and DBUA perform a pre-upgrade check to ensure these users or roles do not exist in the database. Oracle recommends that you do not use these names in your databases. If these users or roles exist, then you should rename or drop them as appropriate before upgrading to Oracle Database 12*c*.

> **See Also:**
>
> *Oracle Database Security Guide* for information on configuring privilege and role authorization for database security

Oracle Update Batching Batch Size Settings Disabled

Oracle update batching settings are disabled in Oracle Database 12c release 2 (12.2). Use JDBC batching instead.

Oracle update batching was deprecated in Oracle Database 12c Release 1 (12.1). Starting in Oracle Database 12c Release 2 (12.2), Oracle update batching is a no operation code (no-op). This means that if you implement Oracle update batching in your application using the Oracle Database 12c Release 2 (12.2) JDBC driver, then the specified batch size is not set, and results in a batch size of 1. With this batch setting, your application processes one row at a time. Oracle strongly recommends that you use the standard JDBC batching if you are using the Oracle Database 12c Release 2 (12.2) JDBC driver.

About Upgrading Tables Dependent on Oracle-Maintained Types

Starting with Oracle Database 12*c* release 2 (12.2), you can run the Parallel Upgrade Utility with the `-T` option to set tables to READ ONLY.

When you run the Parallel Upgrade Utility with the `-T` option, any tablespaces that do not contain Oracle Maintained objects are set to READ ONLY. Setting these tables to READ ONLY can reduce the amount of data that you need to back up before upgrading the database.

If your database has user tables that depend on Oracle Maintained types (for example, AQ queue tables), then you must upgrade these tables manually after upgrade.

After the upgrade is complete, to upgrade tables dependent on Oracle-Maintained types, run the script `utluptabdata.sql` to carry out ALTER TABLE UPGRADE commands on tables in tablespaces set to READ ONLY during the upgrade.

Starting with Oracle Database 12*c* release 2, the ALTER TYPE statement behavior is also changed. If a dependent table is in an accessible tablespace, then it is automatically upgraded to the new version of the type. If the dependent table is in a READ ONLY tablespace, then it is not automatically upgraded. Run the `utluptabdata.sql` script to upgrade those tables set to READ ONLY tablespace states during the upgrade. You only need to run the `utluptabdata.sql` script when you run the Parallel Upgrade Utility with the `-T` option to run the upgrade.

> **Note:**
>
> When tablespaces are set to `READ ONLY`, this setting prevents updates on all tables in the tablespace, regardless of a user's update privilege level. For example, users connecting as SYSDBA are prevented from changing their application data.

Related Topics

- Upgrading Tables Dependent on Oracle-Maintained Types
- Running Upgrades with Read-Only Tablespaces

Case-Insensitive Passwords and ORA-1017 Invalid Username or Password

The Oracle Database 12*c* release 2 (12.2) default authentication protocol is `12` (Exclusive Mode). This protocol requires case-sensitive passwords for authentication. Review your options if you have earlier release password versions.

Starting with Oracle Database 12*c* release 2 (12.2), the default value for the SQLNET.ORA parameter ALLOWED_LOGON_VERSION_SERVER is changed to `12`. This parameter refers to the logon authentication protocol used for the server, not the Oracle Database release.

By default, Oracle no longer supports case-insensitive password-based authentication; only the new password versions (`11G` and `12C`) are allowed. The case-insensitive `10G` password version is no longer generated.

If the following conditions are true, then you may have accounts that are prevented from logging into the database after upgrading to 12.2:

- You are upgrading a server that has user accounts created in an earlier Oracle Database release.
- User accounts created in the earlier release use a case-insensitive password version from an earlier release authentication protocol, such as the `10G` password version.
- Earlier release user accounts have not reset passwords.
- The server has been configured with SEC_CASE_SENSITIVE_LOGON set to FALSE, so that it can only authenticate users who have a `10G` case-insensitive password version.

If you have accounts that require `10G` password versions, then to prevent accounts using that password version from being locked out of the database, you can change from an Exclusive Mode to a more permissive authentication protocol.

> **Note:**
>
> Oracle does not support case-insensitive password-based authentication while running in an Exclusive Mode. The default authentication protocol in Oracle Database 12*c* release 2 (12.2) is an Exclusive Mode. Oracle only supports case-insensitive authentication with the following conditions:
>
> - The server is running in a mode other than an Exclusive Mode
> - The `10G` password version is present

Option for Servers with Accounts Using Only 10G Password Version

After you upgrade to Oracle Database 12*c* release 2 (12.2), complete the following procedure to enable accounts using the `10G` password version:

1. Log in as an administrator.

2. Edit the SQLNET.ORA file to change the SQLNET.ALLOWED_LOGON_VERSION_SERVER setting from the default, 12, to 11 or lower. For example:

   ```
   SQLNET.ALLOWED_LOGON_VERSION_SERVER=11
   ```

After you change to a more permissive SQLNET.ALLOWED_LOGON_VERSION_SERVER setting, expire users' passwords to require them to change their passwords. For detailed information, refer to *Oracle Database Security Guide*.

About Deploying Oracle Grid Infrastructure Using Oracle Fleet Patching and Provisioning

Oracle Fleet Patching and Provisioning (Oracle FPP) is a software lifecycle management method for provisioning and maintaining Oracle homes. Oracle Fleet Patching and Provisioning enables mass deployment and maintenance of standard operating environments for databases, clusters, and user-defined software types.

> **Note:**
>
> Starting with Oracle Grid Infrastructure 19c, the feature formerly known as Rapid Home Provisioning (RHP) is now Oracle Fleet Patching and Provisioning (Oracle FPP).

Oracle Fleet Patching and Provisioning enables you to install clusters, and provision, patch, scale, and upgrade Oracle Grid Infrastructure, Oracle Restart, and Oracle Database homes. The supported versions are 11.2, 12.1, 12.2, 18c, and 19c. You can also provision applications and middleware using Oracle Fleet Patching and Provisioning.

Oracle Fleet Patching and Provisioning is a service in Oracle Grid Infrastructure that you can use in either of the following modes:

- Central Oracle Fleet Patching and Provisioning Server

  The Oracle Fleet Patching and Provisioning Server stores and manages standardized images, called gold images. Gold images can be deployed to any number of nodes across the data center. You can create new clusters and databases on the deployed homes and can use them to patch, upgrade, and scale existing installations.

  The Oracle Fleet Patching and Provisioning Server can manage the following types of installations:

  – Software homes on the cluster hosting the Oracle Fleet Patching and Provisioning Server itself.
  – Oracle Fleet Patching and Provisioning Clients running Oracle Grid Infrastructure 12c Release 2 (12.2), 18c, and 19c.
  – Installations running Oracle Grid Infrastructure 11g Release 2 (11.2) and 12c Release 1 (12.1).
  – Installations running without Oracle Grid Infrastructure.

The Oracle Fleet Patching and Provisioning Server can provision new installations and can manage existing installations without requiring any changes to the existing installations. The Oracle Fleet Patching and Provisioning Server can automatically share gold images among peer servers to support enterprises with geographically distributed data centers.

- Oracle Fleet Patching and Provisioning Client

  The Oracle Fleet Patching and Provisioning Client can be managed from the Oracle Fleet Patching and Provisioning Server, or directly by executing commands on the client itself. The Oracle Fleet Patching and Provisioning Client is a service built into the Oracle Grid Infrastructure and is available in Oracle Grid Infrastructure 12c Release 2 (12.2) and later releases. The Oracle Fleet Patching and Provisioning Client can retrieve gold images from the Oracle Fleet Patching and Provisioning Server, upload new images based on the policy, and apply maintenance operations to itself.

Oracle Fleet Patching and Provisioning

Deploying Oracle software using Oracle Fleet Patching and Provisioning has the following advantages:

- Ensures standardization and enables high degrees of automation with gold images and managed lineage of deployed software.
- Minimizes downtime by deploying new homes as images (called gold images) out-of-place, without disrupting active databases or clusters.
- Simplifies maintenance by providing automatons which are invoked with a simple, consistent API across database versions and deployment models.
- Reduces maintenance risk with built-in validations and a “dry run” mode to test the operations.
- Enables you to resume or restart the commands in the event of an unforeseen issue, reducing the risk of maintenance operations.
- Minimizes and often eliminates the impact of patching and upgrades, with features that include:
  - Zero-downtime database upgrade with fully automated upgrade, executed entirely within the deployment without requiring any extra nodes or external storage.
  - Adaptive management of database sessions and OJVM during rolling patching.
  - Options for management of consolidated deployments.
- The deployment and maintenance operations enable customizations to include environment-specific actions into the automated workflow.

Related Topics

- *Oracle Clusterware Administration and Deployment Guide*

> **See Also:**
>
> *Oracle Clusterware Administration and Deployment Guide* for information about setting up the Oracle Fleet Patching and Provisioning Server and Client, and for creating and using gold images for provisioning and patching Oracle Grid Infrastructure and Oracle Database homes.

Restrictions Using Zero Data Loss Recovery Appliance Release 12.1 Backups

Zero Data Loss Recovery Appliance release 12.1 does not support backups from protected database clients using Oracle Database 12*c* release 2 (12.2).

Zero Data Loss Recovery Appliance release 12.2 (Recovery Appliance) does support backups from protected release 12.2 database clients.

If you back up your database to Recovery Appliance, then Oracle recommends that you do not not upgrade your database to release 12.2 until your Recovery Appliance is upgraded to release 12.2.

Client and Foreground Server Process Memory Changes

To increase optimization and performance, the Oracle Database Client and server process memory requirements are greater than in previous releases.

Every release of Oracle Database includes new features and capabilities. To provide optimal performance for the increased capability of the database, there can be an increase in the Oracle Database Client and Oracle Database Server can increase from one release to the next. The memory requirement increase can vary from platform to platform.

As part of your upgrade plan, check to determine the memory requirements increase that can be present in a new Oracle Database release. For example, in comparison to Oracle Database 11g Release 2 (11.2), Oracle Database 12c on some platforms can have as much as a 5 MB memory increase for each client, and a 10 MB increase for each server.

Deprecated Initialization Parameters in Oracle Database 12c Release 2 (12.2)

To understand changes and replacements in parameter settings, review the parameters deprecated in the 12.2 release. These parameters can be removed in a later release.

O7_DICTIONARY_ACCESSIBILITY Initialization parameter

The initialization parameter `O7_DICTIONARY_ACCESSIBILITY` controls restrictions on `SYSTEM` privileges. If the parameter is set to `TRUE`, then access to objects in the SYS schema is allowed. The default setting is FALSE. This default setting prevents system privileges that allow access to objects in any schema from allowing access to objects in the SYS schema. The `O7_DICTIONARY_ACCESSIBILITY` parameter is deprecated.

ASM_PREFERRED_READ_FAILURE_GROUPS Initialization Parameter

The `ASM_PREFERRED_READ_FAILURE_GROUPS` initialization parameter is deprecated in Oracle Automatic Storage Management 12c release 2 (12.2.0.1). Starting with Oracle Automatic Storage Management (Oracle ASM) 12c release 2 (12.2.0.1), specifying the preferred read failure groups is done automatically, so the use of the `ASM_PREFERRED_READ_FAILURE_GROUPS` initialization parameter is no longer required. Use the PREFERRED_READ.ENABLED disk group attribute to control the preferred read functionality.

PARALLEL_ADAPTIVE_MULTI_USER Initialization Parameter

The initialization parameter `PARALLEL_ADAPTIVE_MULTI_USER` specifies if you want to use an adaptive algorithm to improve performance in multi-user environments that use parallel execution. This parameter is deprecated, and the default value is now `FALSE`. There is no replacement for this parameter. Oracle recommends that you use the Oracle Database feature Parallel Statement Queuing to obtain parallel execution performance gains.

UTL_FILE_DIR Initialization Parameter

The initialization parameter `UTL_FILE_DIR` specifies accessible directories for PL/SQL file I/O. This parameter is deprecated, and Oracle recommends that you do not provide `UTL_FILE_DIR` access. Oracle recommends that you instead use the directory object feature, which replaces `UTL_FILE_DIR`. Directory objects provide the following benefits:

- They offer more flexibility and granular control to the `UTL_FILE` application administrator
- They can be maintained dynamically, without shutting down the database
- They are consistent with other Oracle tools.

Related Topics

- *Oracle Database Reference*

Deprecated Features in Oracle Database 12c Release 2 (12.2)

Review the deprecated features listed in this section to prepare to use alternatives after you upgrade.

- Deprecation of ALTER TYPE REPLACE
  Starting with Oracle Database 12*c* release 2 (12.2.0.1), the REPLACE clause of ALTER TYPE is deprecated.
- Deprecation of configToolAllCommands Script
  The postinstallation check script configToolAllCommands is deprecated in Oracle Database 12c release 1 (12.1).
- Deprecation of DBMS_DEBUG Package
  The DBMS_DEBUG package is deprecated in Oracle Database 12*c* release 2 (12.2). Oracle recommends that you use DBMS_DEBUG_JDWP.
- Deprecation of Intelligent Data Placement (IDC)
  Intelligent Data Placement is deprecated in Oracle Database 12c release 2 (12.2).

- Deprecation of CONTINUOUS_MINE Option
  Starting with Oracle Database 12c Release 2 (12.2.0.1), the LogMiner CONTINUOUS_MINE option is deprecated.
- Deprecation of Non-CDB Architecture
  The non-CDB architecture was deprecated in Oracle Database 12c. It can be desupported and unavailable in a release after Oracle Database 19c .
- Deprecation of Oracle Administration Assistant for Windows
  Oracle Administration Assistant for Windows is deprecated in Oracle Database 12*c* release 2 (12.2).
- Deprecation of Oracle Data Provider for .NET PromotableTransaction Setting
  The Oracle Data Provider for .NET `PromotableTransaction` setting is deprecated, because it is no longer necessary.
- Deprecation of oracle.jdbc.OracleConnection.unwrap()
  Starting in Oracle Database 12c release 2 (12.2), the Java package `oracle.jdbc.OracleConnection.unwrap()` is deprecated.
- Deprecation of oracle.jdbc.rowset Package
  Starting in Oracle Database 12*c* release 2 (12.2), the Java `oracle.jdbc.rowset` package is deprecated
- Deprecation of oracle.sql.DatumWithConnection Classes
  `oracle.sql` classes that extend `oracle.sql.DatumWithConnection` are deprecated in Oracle Database 12*c* release 2 (12.2), in favor of `oracle.jdbc` extension types.
- Deprecation of Oracle Multimedia Java APIs
  The Oracle Multimedia Java APIs are deprecated in Oracle Database 12c release 2.
- Deprecation of Oracle Multimedia Support for DICOM
  Starting in Oracle Database 12c release 2 (12.2), the Oracle Multimedia DICOM feature is deprecated.
- Deprecation of Multimedia SQL/MM Still Image Standard Support
  Starting in Oracle Database 12c release 2 (12.2), Oracle Multimedia SQL/MM Still Image standard support is deprecated.
- Deprecation of Unicode Collation Algorithm (UCA) 6.1 Collations
  Starting in Oracle Database 12*c* release 2, the Unicode Collation Algorithm (UCA) 6.1 collations are deprecated.
- Deprecation of UNIFIED_AUDIT_SGA_QUEUE_SIZE
  Starting in Oracle Database 12*c* release 2, the initialization parameter UNIFIED_AUDIT_SGA_QUEUE_SIZE is deprecated.
- Deprecation of VERIFY_FUNCTION and VERIFY_FUNCTION_11G
  The `VERIFY_FUNCTION` and `VERIFY_FUNCTION_11G` password verify functions are deprecated in this release, because they enforce the weaker password restrictions from earlier releases.
- Deprecation of V$MANAGED_STANDBY
  The V$MANAGED_STANDBY view is deprecated in Oracle Database 12*c* release 2 (12.2.0.1). Oracle recommends that you use the new view V$DATAGUARD_PROCESS.
- Deprecation of Some XML DB Functions
  Starting with Oracle Database 12*c* release 2 (12.2) the options listed in this topic are deprecated.

- Deprecated Features for Oracle XML Database
  These features are deprecated in Oracle Database 12c Release 1, and can be desupported in a future release.

Deprecation of ALTER TYPE REPLACE

Starting with Oracle Database 12*c* release 2 (12.2.0.1), the REPLACE clause of ALTER TYPE is deprecated.

As an alternative, Oracle recommends that you use the ALTER TYPE methods ADD and DROP, or use ALTER TYPE *method* ADD .

Related Topics

- *Oracle Database PL/SQL Language Reference*

Deprecation of configToolAllCommands Script

The postinstallation check script configToolAllCommands is deprecated in Oracle Database 12c release 1 (12.1).

The script `configToolAllCommands` runs in the response file mode to configure Oracle products after installation. It uses a separate password response file. Starting with Oracle Database 12c release 2 (12.2), `configToolAllCommands` is deprecated. It may be desupported in a future release.

You can now obtain postinstallation checks as part of the installation process. Oracle recommends that you run the Oracle Database or Oracle Grid Infrastructure installer with the option `-executeConfigTools`. You can use the same response file created during installation to complete postinstallation configuration.

Deprecation of DBMS_DEBUG Package

The DBMS_DEBUG package is deprecated in Oracle Database 12*c* release 2 (12.2). Oracle recommends that you use DBMS_DEBUG_JDWP.

In earlier releases, PL/SQL included the DBMS_DEBUG package to enable internal and third-party tools to debug PL/SQL programs. The DBMS_DEBUG package provides APIs to set breakpoints, obtain values of variables, and so on. This functionality has been provided by the DBMS_DEBUG_JDWP package for several releases. DBMS_DEBUG_JDWP provides the equivalent PL/SQL debugging capabilities, and it enables seamless debugging of PL/SQL routines when it calls into or is called from server-side Java (OJVM) with Java stored procedures.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

Deprecation of Intelligent Data Placement (IDC)

Intelligent Data Placement is deprecated in Oracle Database 12c release 2 (12.2).

Intelligent Data Placement enables you to specify disk regions on Oracle ASM disks for best performance. Using the disk region settings, you can ensure that frequently accessed data is placed on the outermost (hot) tracks which have greater speed and higher bandwidth. In addition, files with similar access patterns are located physically

close, reducing latency. Intelligent Data Placement also enables the placement of primary and mirror extents into different hot or cold regions

This feature is deprecated in Oracle Database 12c release 2 (12.2).

Related Topics

- *Oracle Automatic Storage Management Administrator's Guide*

Deprecation of CONTINUOUS_MINE Option

Starting with Oracle Database 12c Release 2 (12.2.0.1), the LogMiner CONTINUOUS_MINE option is deprecated.

The LogMiner `CONTINUOUS_MINE` option is still supported for backward compatibility reasons. However, Oracle recommends that you discontinue using it. There is no replacement functionality.

Deprecation of Non-CDB Architecture

The non-CDB architecture was deprecated in Oracle Database 12c. It can be desupported and unavailable in a release after Oracle Database 19c .

Oracle recommends use of the CDB architecture.

Deprecation of Oracle Administration Assistant for Windows

Oracle Administration Assistant for Windows is deprecated in Oracle Database 12*c* release 2 (12.2).

Oracle Administration Assistant for Windows is a tool for creating database administrators, operators, users, and roles in Windows. It also allows database service, startup and shutdown configuration, and Windows Registry parameter management.

Instead of using Oracle Administration Assistant for Windows, use native Windows administration tools.

Deprecation of Oracle Data Provider for .NET PromotableTransaction Setting

The Oracle Data Provider for .NET `PromotableTransaction` setting is deprecated, because it is no longer necessary.

Promotable transactions themselves are not being deprecated. Only this specific setting is deprecated.

The Oracle Data Provider for .NET registry setting `PromotableTransaction` indicates whether the application must keep transactions as local, or if it can begin all single connection transactions as local, and then promote the transaction to distributed when a second connection enlists. This is the concept of promotable transactions.

The Promotable Transaction setting is deprecated in Oracle Database 12*c* release 2 (12.2). There is no reason not to use promotable transactions. Oracle recommends you accept the default value `promotable`.

Deprecation of oracle.jdbc.OracleConnection.unwrap()

Starting in Oracle Database 12c release 2 (12.2), the Java package `oracle.jdbc.OracleConnection.unwrap()` is deprecated.

The Java package `oracle.jdbc.OracleConnection.unwrap()` is deprecated in Oracle Database 12c release 2, and later releases. There is no replacement for this package.

Oracle recommends that you replace this JDBC method in your applications with standard Java methods.

Related Topics

- https://support.oracle.com/rs?type=doc&id=2024500.1

Deprecation of oracle.jdbc.rowset Package

Starting in Oracle Database 12*c* release 2 (12.2), the Java `oracle.jdbc.rowset` package is deprecated

Oracle recommends that you use the Standard JDBC RowSet package to replace this feature.

Related Topics

- *Oracle Database JDBC Developer's Guide*
- https://support.oracle.com/rs?type=doc&id=2024500.1

Deprecation of oracle.sql.DatumWithConnection Classes

`oracle.sql` classes that extend `oracle.sql.DatumWithConnection` are deprecated in Oracle Database 12*c* release 2 (12.2), in favor of `oracle.jdbc` extension types.

In previous releases, Oracle Database included Oracle JDBC drivers that provided specific type extensions and performance extensions in both `oracle.sql` and `oracle.jdbc` Java packages. Starting with Oracle Database 12*c* release 2 (12.2), the `oracle.sql` classes that extend `oracle.sql.DatumWithConnection` are deprecated. The `oracle.jdbc` extensions continue to be supported.

For example, here is a partial list of deprecated `oracle.sql` classes:

- ARRAY
- BFILE
- BLOB
- CLOB
- OPAQUE
- REF
- STRUCT

Oracle recommends that you replace `oracle.sql` classes that extend `oracle.sql.DatumWithConnection` in your applications with standard Java types, or with `oracle.jdbc` extensions.

Deprecation of Oracle Multimedia Java APIs

The Oracle Multimedia Java APIs are deprecated in Oracle Database 12c release 2.

The following Java APIs are deprecated in Oracle Database 12c Release 2 (12.2), and can be desupported in a future release:

- Oracle Multimedia Java API
- Oracle Multimedia Servlets and JSP Java API
- Oracle Multimedia DICOM Java API
- Oracle Multimedia Mid-Tier Java API

Deprecation of Oracle Multimedia Support for DICOM

Starting in Oracle Database 12c release 2 (12.2), the Oracle Multimedia DICOM feature is deprecated.

There is no replacement for DICOM support in Oracle Database.

Deprecation of Multimedia SQL/MM Still Image Standard Support

Starting in Oracle Database 12c release 2 (12.2), Oracle Multimedia SQL/MM Still Image standard support is deprecated.

For image processing operations, Oracle Multimedia developers can call the new ORD_IMAGE PL/SQL package, or call the ORDImage methods.

For image matching, Oracle Database developers can use open source packages, such as OpenCV.

Deprecation of Unicode Collation Algorithm (UCA) 6.1 Collations

Starting in Oracle Database 12*c* release 2, the Unicode Collation Algorithm (UCA) 6.1 collations are deprecated.

The Unicode Collation Algorithm (UCA) 6.1 collations (UCA0610_*) are deprecated. They can be desupported and unavailable in a future release. Oracle recommends that you use the latest supported version of UCA collations for sorting multilingual data.

Related Topics

- *Oracle Database Globalization Support Guide*

Deprecation of UNIFIED_AUDIT_SGA_QUEUE_SIZE

Starting in Oracle Database 12*c* release 2, the initialization parameter UNIFIED_AUDIT_SGA_QUEUE_SIZE is deprecated.

The UNIFIED_AUDIT_SGA_QUEUE_SIZE parameter is deprecated, and the value for this parameter is no longer honored. However, the parameter is currently retained for backward compatibility.

See *Oracle Database Security Guide* for additional information about Unified Audit records.

Related Topics

- *Oracle Database Security Guide*

Deprecation of VERIFY_FUNCTION and VERIFY_FUNCTION_11G

The `VERIFY_FUNCTION` and `VERIFY_FUNCTION_11G` password verify functions are deprecated in this release, because they enforce the weaker password restrictions from earlier releases.

Oracle recommends that you use the functions `ORA12C_VERIFY_FUNCTION` and `ORA12C_STRONG_VERIFY_FUNCTION`. These functions enforce stronger, more up-to-date password verification restrictions.

> **See Also:**
>
> *Oracle Database Security Guide*

Deprecation of V$MANAGED_STANDBY

The V$MANAGED_STANDBY view is deprecated in Oracle Database 12*c* release 2 (12.2.0.1). Oracle recommends that you use the new view V$DATAGUARD_PROCESS.

The V$DATAGUARD_PROCESS view includes much more information about processes used by Oracle Data Guard.

Related Topics

- *Oracle Database Reference*

Deprecation of Some XML DB Functions

Starting with Oracle Database 12*c* release 2 (12.2) the options listed in this topic are deprecated.

The following options are deprecated:

- Oracle XQuery function `ora:contains`. Use XQuery Full Text instead.
- Oracle SQL function XMLRoot. Use SQL/XML function `XMLSerialize()` with a version number instead.
- Nested tables stored as index-ordered tables (IOTs). This includes both the use of option `DBMS_XMLSCHEMA.REGISTER_NT_AS_IOT`, and the use of clause `NESTED TABLE N STORE AS ... (ORGANIZATION INDEX)` when creating a table with nested-table column *N*. Instead, store nested-table columns using heap storage (the default behavior for PL/SQL procedure `DBMS_XMLSCHEMA.registerSchema`).
- PL/SQL procedure `DBMS_XSLPROCESSOR.CLOB2FILE`. Use `DBMS_LOB.CLOB2FILE` instead.
- PL/SQL function `DBMS_XSLPROCESSOR.READ2CLOB`. Use `DBMS_LOB.LOADCLOBFROMFILE` instead.
- Use of XLink with Oracle XML DB.

- Oracle XML DB Content Connector.

For more information, refer to *Oracle XML DB Developer's Guide*.

Related Topics

- *Oracle XML DB Developer's Guide*

Deprecated Features for Oracle XML Database

These features are deprecated in Oracle Database 12c Release 1, and can be desupported in a future release.

- `CLOB` storage of `XMLType`, also known as unstructured storage, is deprecated. Use binary XML storage of `XMLType` instead.

  To preserve whitespace in an XML file, store two copies of your original XML document. Use one file as an `XMLType` instance for database use and XML processing, and use the other file as a `CLOB` instance to provide document fidelity.
- Creating an `XMLIndex` index over an XML fragment stored as a `CLOB` instance embedded in object-relational `XMLType` data is deprecated. If you must index the data in such a fragment, then store the document using binary XML storage, instead of object-relational storage.
- The following PL/SQL subprograms in package `DBMS_XMLSCHEMA` are deprecated:
  - `generateSchema`
  - `generateSchemas`

  There are no replacements for these constructs, and there is no workaround for this change.
- PL/SQL package `DBMS_XDB_CONFIG` is new. All Oracle XML((nbsp))DB configuration functions, procedures, and constants are moved from package `DBMS_XDB` to `DBMS_XDB_CONFIG`. These functions, procedures and constants are now deprecated for package `DBMS_XDB`. Use them in package `DBMS_XDB_CONFIG` instead.

  The following is a list of subprograms deprecated in package `DBMS_XDB`:
  - `ADDHTTPEXPIREMAPPING`
  - `ADDMIMEMAPPING`
  - `ADDSCHEMALOCMAPPING`
  - `ADDSERVLET`
  - `ADDSERVLETMAPPING`
  - `ADDSERVLETSECROLE`
  - `ADDXMLEXTENSION`
  - `CFG_GET`
  - `CFG_REFRESH`
  - `CFG_UPDATE`
  - `DELETEHTTPEXPIREMAPPING`
  - `DELETEMIMEMAPPING`

  - DELETESCHEMALOCMAPPING
  - DELETESERVLET
  - DELETESERVLETMAPPING
  - DELETESERVLETSECROLE
  - DELETEXMLEXTENSION
  - GETFTPPORT
  - GETHTTPPORT
  - GETLISTENERENDPOINT
  - SETFTPPORT
  - SETHTTPPORT
  - SETLISTENERENDPOINT
  - SETLISTENERLOCALACCESS

  The following is a list of constants that are deprecated in package DBMS_XDB:

  - XDB_ENDPOINT_HTTP
  - XDB_ENDPOINT_HTTP2
  - XDB_PROTOCOL_TCP
  - XDB_PROTOCOL_TCPS

- All Oracle SQL functions for updating XML data are deprecated. Use XQuery Update instead for these functions . The following is a list of deprecated XML updating functions:
  - updateXML
  - insertChildXML
  - insertChildXMLbefore
  - insertChildXMLafter
  - insertXMLbefore
  - insertXMLafter
  - appendChildXML
  - deleteXML
- Oracle SQL function sys_xmlgen is deprecated. Use the SQL/XML generation functions instead.
- The following Oracle XQuery functions are deprecated. Use the corresponding standard XQuery functions instead, that is, the functions with the same names but with namespace prefix fn.
  - ora:matches – use fn:matches instead
  - ora:replace – use fn:replace instead
- The following Oracle constructs that provide support for XML translations are deprecated.
  - PL/SQL package DBMS_XMLTRANSLATIONS

  - Oracle XPath function `ora:translate`
  - XML Schema annotations `xdb:maxOccurs`, `xdb:srclang`, and `xdb:translate`

  There are no replacements for these constructs, and there is no workaround for this change.

- The following XML Schema annotations are deprecated:
  - `xdb:defaultTableSchema`
  - `xdb:maintainOrder`
  - `xdb:mapUnboundedStringToLob`
  - `xdb:maxOccurs`
  - `xdb:SQLCollSchema`
  - `xdb:SQLSchema`
  - `xdb:srclang`
  - `xdb:storeVarrayAsTable`
  - `xdb:translate`

  There are no replacements for these constructs, and there is no workaround for this change.
- The value `xml_clobs` for export parameter `data_options` is deprecated starting with Oracle Database 12c.

Desupported Initialization Parameters in Oracle Database 12c Release 2 (12.2)

Review this list of desupported initialization parameters for changes and replacements in parameter settings in this release.

GLOBAL_CONTEXT_POOL_SIZE Initialization Parameter

The `GLOBAL_CONTEXT_POOL_SIZE` initialization parameter is removed and desupported in this release.

`GLOBAL_CONTEXT_POOL_SIZE` specified the amount of memory to allocate in the SGA for storing and managing global application context. The default value of this parameter was null. The parameter was deprecated in Oracle Database 10g release 2 (10.2).

MAX_ENABLED_ROLES Initialization Parameter

The `MAX_ENABLED_ROLES` initialization parameter is removed and desupported in this release.

There is no replacement for this parameter. Oracle Database has not used this parameter since Oracle Database 10g release 2 (10.2).

OPTIMIZER_ADAPTIVE_FEATURES Initialization Parameter

The `OPTIMIZER_ADAPTIVE_FEATURES` initialization parameter is removed and desupported in this release.

The functions of this parameter are replaced by two new parameters. The default value for OPTIMIZER_ADAPTIVE_PLANS is TRUE. When set to TRUE, this parameter determines alternate execution plans that are based on statistics collected as a query executes. OPTIMIZER_ADAPTIVE_STATISTICS is set by default to FALSE. When set to TRUE, the optimizer augments the statistics gathered in the database with adaptive statistics gathered at SQL statement parse time to improve the quality of SQL execution plans.

PARALLEL_AUTOMATIC_TUNING Initialization Parameter

The PARALLEL_AUTOMATIC_TUNING initialization parameter is removed and desupported in this release.

The PARALLEL_AUTOMATIC_TUNING initialization parameter determined the default values for parameters that controlled parallel processing. It was deprecated in Oracle Database 10g release 2 (10.2).

PARALLEL_IO_CAP_ENABLED Initialization Parameter

The PARALLEL_IO_CAP_ENABLED initialization parameter determined if Oracle Database set a limit to the default degree of parallelism to a level no greater than the I/O system supported. This parameter was deprecated in Oracle Database release 11.2. The function of this parameter was replaced by the PARALLEL_DEGREE_LIMIT parameter, when that parameter is set to IO.

PARALLEL_SERVER Initialization Parameter

The PARALLEL_SERVER initialization parameter is removed and desupported in this release.

The PARALLEL_SERVER initialization parameter was used to start a database in Oracle Parallel Server mode. This parameter was deprecated in Oracle9i Database Release 1 (9.0.1). Oracle Parallel Server was replaced with Oracle Real Application Clusters, which uses the CLUSTER_DATABASEinitialization parameter.

PARALLEL_SERVER_INSTANCES Initialization Parameter

The PARALLEL_SERVER_INSTANCES initialization parameter is removed and desupported in this release.

The PARALLEL_SERVER_INSTANCES initialization parameter specified the number of configured instances in Oracle Parallel Server mode. This parameter was deprecated in Oracle9i Database Release 1 (9.0.1). Oracle Parallel Server was replaced with Oracle Real Application Clusters, which uses the CLUSTER_DATABASE_INSTANCES initialization parameter.

USE_INDIRECT_DATA_BUFFERS Initialization Parameter

The initialization parameter USE_INDIRECT_DATA_BUFFERS is removed and desupported in this release.

The parameter was used to enable the Very Large Memory feature for 32-bit platforms. These platforms are no longer supported.

Related Topics

- *Oracle Database Reference*

Desupported Features in Oracle Database 12c Release 2 (12.2)

Review this list of desupported features as part of your upgrade planning.

- Desupport of Advanced Replication
  Starting in Oracle Database 12c release 2 (12.2), the Advanced Replication feature of Oracle Database is desupported.
- Desupport of Direct File System Placement for OCR and Voting Files
  Placing OCR and Voting Disk files on shared file systems is desupported in favor of placing the files on Oracle ASM.
- Desupport of JPublisher
  All Oracle JPublisher features are desupported and unavailable in Oracle Database 12*c* Release 2 (12.2.0.1).
- Desupport of preupgrd.sql and utluppkg.sql
  The `preupgrd.sql` and `utluppkg.sql` scripts are replaced by the Preupgrade Information Tool (`preupgrade.jar`).
- Desupported Oracle Data Provider for .NET APIs for Transaction Guard
  Application programming interfaces (APIs) for Transaction Guard listed here are desupported in Oracle Database 12*c* release 2 (12.2).
- Desupported Views in Oracle Database 12c Release 2 (12.2)
  The views listed in this topic are desupported in Oracle Database 12*c* release 2 (12.2).
- SQLJ Support Inside Oracle Database
  Starting with Oracle Database 12*c* release 2 (12.2), Oracle does not support running server-side SQLJ code.
- Desupport of Some XML DB Features
  Starting in Oracle Database 12*c* release 2 (12.2), the XML DB features listed here are desupported.
- Desupport of MAX_SHARED_TEMP_SIZE
  The `CREATE_PLUGGABLE_DATABASE` statement `pdb_storage_clause` `MAX_SHARED_TEMP_SIZE` parameter is desupported in Oracle Database 12c Release 2 (12.2). Use `MAXSIZE` instead.

Desupport of Advanced Replication

Starting in Oracle Database 12c release 2 (12.2), the Advanced Replication feature of Oracle Database is desupported.

The Oracle Database Advanced Replication feature is desupported in its entirety. The desupport of this feature includes all functionality associated with this feature: multimaster replication, updateable materialized views, and deployment templates. Read-only materialized views are still supported with basic replication.

Oracle recommends that you replace your use of Advanced Replication with Oracle GoldenGate.

Desupport of Direct File System Placement for OCR and Voting Files

Placing OCR and Voting Disk files on shared file systems is desupported in favor of placing the files on Oracle ASM.

Starting with Oracle Grid Infrastructure 12c Release 2 (12.2), the placement of Oracle Clusterware files: the Oracle Cluster Registry (OCR), and the Voting Files, directly on a shared file system is desupported in favor of having Oracle Clusterware files managed by Oracle Automatic Storage Management (Oracle ASM). You cannot place Oracle Clusterware files directly on a shared file system. If you need to use a supported shared file system, either a Network File System, or a shared cluster file system instead of native disks devices, then you must create Oracle ASM disks on supported network file systems that you plan to use for hosting Oracle Clusterware files before installing Oracle Grid Infrastructure. You can then use the Oracle ASM disks in an Oracle ASM disk group to manage Oracle Clusterware files.

If your Oracle Database files are stored on a shared file system, then you can continue to use the same for database files, instead of moving them to Oracle ASM storage.

Desupport of JPublisher

All Oracle JPublisher features are desupported and unavailable in Oracle Database 12*c* Release 2 (12.2.0.1).

Oracle recommends that you use the following alternatives:

- To continue to use Web service callouts, Oracle recommends that you use the Oracle JVM Web Services Callout utility, which is a replacement for the Web Services Callout utility.
- To replace other JPublisher automation capabilities, including mapping user-defined SQL types or SQL types, wrapping PL/SQL packages and similar capabilities, Oracle recommends that developers use explicit steps, such as precompiling code with SQLJ precompiler, building Java STRUCT classes, or using other prestructured options.

Related Topics

- https://support.oracle.com/rs?type=doc&id=1937939.1

> **See Also:**
>
> My Oracle Support Note 1937939.1 for more information about JDeveloper deprecation and desupport

Desupport of preupgrd.sql and utluppkg.sql

The `preupgrd.sql` and `utluppkg.sql` scripts are replaced by the Preupgrade Information Tool (`preupgrade.jar`).

Beginning with Oracle Database 12c release 2 (12.2), the Pre-Upgrade Information Tool scripts `preupgrd.sql` and `utluppkg.sql` are no longer supplied as part of

the Oracle Database release. The Pre-Upgrade Information Tool `preupgrade.jar` replaces both of these files.

The `preupgrade.jar` Pre-Upgrade Information Tool is supplied with Oracle Database 12c release 2 (12.2). This script has the same capabilities as the scripts it replaces. It can run using the Java Development Kits (JDKs) installed with Oracle Database releases supported for direct upgrade to Oracle Database 12c release 2 (12.2).

Desupported Oracle Data Provider for .NET APIs for Transaction Guard

Application programming interfaces (APIs) for Transaction Guard listed here are desupported in Oracle Database 12*c* release 2 (12.2).

The following Oracle Data Provider for .NET application programming interfaces for Transaction Guard are desupported in Oracle Database 12*c* Release 2 (12.2):

- `OracleLogicalTransactionStatus` class
- `OracleConnection.GetLogicalTransactionStatus` method
- `OracleConnection.LogicalTransactionId` property
- `OracleConnection.OracleLogicalTransaction` property
- `OracleLogicalTransaction.DataSource` property
- `OracleLogicalTransaction.GetOutcome()` method
- `OracleLogicalTransaction.GetOutcome(string, string, string)` method
- `OracleLogicalTransaction.UserId` property

Desupported Views in Oracle Database 12c Release 2 (12.2)

The views listed in this topic are desupported in Oracle Database 12*c* release 2 (12.2).

Revise any of your SQL statements that use these views.

DBA_REGISTERED_MVIEW_GROUPS View

V$REPLPROP View

V$REPLQUEUE View

SQLJ Support Inside Oracle Database

Starting with Oracle Database 12*c* release 2 (12.2), Oracle does not support running server-side SQLJ code.

Oracle supports using client-side SQLJ. However, Oracle does not support the use of server-side SQLJ, including running stored procedures, functions, and triggers in the database environment.

Desupport of Some XML DB Features

Starting in Oracle Database 12*c* release 2 (12.2), the XML DB features listed here are desupported.

The following features are desupported:

- Java classes in package `oracle.xdb.dom`
- Oracle XPath function `ora:instanceof`. Use XQuery operator `instance of` instead.
- Oracle XPath function `ora:instanceof-only`. Use XML Schema attribute `xsi:type` instead.
- Function-based indexes on XMLType. Use XMLIndex with a structured component instead.
- Oracle XQuery function `ora:view`. Use XQuery functions `fn:collection` instead.
- PL/SQL procedure `DBMS_XDB_ADMIN.CreateRepositoryXMLIndex`
- PL/SQL procedure `DBMS_XDB_ADMIN.XMLIndexAddPath`
- PL/SQL procedure `DBMS_XDB_ADMIN.XMLIndexRemovePath`
- PL/SQL procedure `DBMS_XDB_ADMIN.DropRepositoryXMLIndex`
- XML schema annotation (attribute) `csx:encodingType`
- XMLIndex index on CLOB portions of hybrid XMLType storage (index on CLOB data that is embedded within object-relational storage)

Desupport of MAX_SHARED_TEMP_SIZE

The `CREATE_PLUGGABLE_DATABASE` statement `pdb_storage_clause` `MAX_SHARED_TEMP_SIZE` parameter is desupported in Oracle Database 12c Release 2 (12.2). Use `MAXSIZE` instead.

The `MAX_SHARED_TEMP_SIZE` parameter of `pdb_storage_clause` was removed in Oracle Database 12c Release 2 (12.2), because it is unneeded. The `MAXSIZE` parameter limits the amount of storage that can be used by all tablespaces in the pluggable database (PDB) to the value specified with `size_clause`. This limit includes the size of data files and temporary files for tablespaces belonging to the PDB. Accordingly MAX_SHARED_TEMP_SIZE has no function.

Behavior Changes, Deprecations and Desupports in Oracle Database 12c Release 1 (12.1)

Review for descriptions of Oracle Database 12c Release 1 (12.1) changes.

- Behavior Changes for Oracle Database 12c Release 1 (12.1)
  Review these behavior changes to help plan for upgrades to Oracle Database 12c Release 1 (12.1).
- Deprecated Features in Oracle Database 12c Release 1 (12.1)
  Review the deprecated features listed in this section to prepare to use alternatives after you upgrade. Features deprecated in an earlier release can be desupported in a later release.
- Deprecated Views in Oracle Database 12c Release 1 (12.1)
  Review the deprecated features listed in this section to prepare to use alternatives after you upgrade. Features deprecated in an earlier release can be desupported in a later release.

- Deprecated Initialization Parameters in Oracle Database 12c Release 1 (12.1)
  To understand changes and replacements in parameter settings, review the parameters deprecated in the 12.1 release. These parameters can be removed in a later release.
- Desupported Features in Oracle Database 12c Release 1(12.1)
  Review this list of desupported features as part of your upgrade planning.

Behavior Changes for Oracle Database 12c Release 1 (12.1)

Review these behavior changes to help plan for upgrades to Oracle Database 12c Release 1 (12.1).

- Error Associated with catupgrd.sql Run Without PARALLEL=NO
  If you choose to run the `catupgrd.sql` script instead of running `catctl.pl`), then you now must provide information for an additional input parameter, PARALLEL.
- Changes for Deinstallation and Cleanup of Oracle Base
  Review these topics to understand deinstallation and Oracle Base cleanup script changes.
- Identifying and Dropping Deprecated and Desupported Parameters
  To locate deprecated parameters, you can use the `SELECT NAME FROM V$PARAMETER` query, and then drop those parameters with `ALTER SYSTEM RESET`.
- Changes to Security Auditing Features
  The full set of auditing features are available automatically in Oracle Database 12c release 1 (12.1) and later releases.
- Upgrading a System that Did Not Have SQLNET.ALLOWED_LOGON_VERSION Parameter Setting
  Review the parameter setting for `SQLNET.ALLOWED_LOGON_VERSION_SERVER` to determine its implications for security and client connections to the upgraded database.
- Oracle Warehouse Builder (OWB) Not Installed with Oracle Database
  Oracle Warehouse Builder must be installed separately.
- About Upgrading Oracle Database Release 10.2 or 11.1 and OCFS and RAW Devices
  If you are upgrading an Oracle Database release 10.2.0.5 or release 11.1.0.7 environment that stores Oracle Clusterware files on OCFS on Windows or RAW devices, then you cannot directly upgrade to Oracle Database 12*c* release 1 (12.1).
- Change to VARCHAR2, NVARCHAR2, and RAW Datatypes
  You can increase the MAX_STRING_SIZE value for these datatypes to `32767` bytes in Oracle Database 12c Release 1 (12.1) and later releases.
- Changed Default for RESOURCE_LIMIT Parameter
  RESOURCE_LIMIT is set to TRUE by default.
- Oracle XML DB is Mandatory and Cannot Be Uninstalled
  Starting with Oracle Database 12c, Oracle XML DB is a mandatory component of Oracle Database.

- Direct NFS Enabled By Default for Oracle RAC
  In Oracle Database 12c release 1 (12.1) and later releases, Direct NFS (DNFS) is enabled by default with Oracle Real Application Clusters (Oracle RAC) installations.

Error Associated with catupgrd.sql Run Without PARALLEL=NO

If you choose to run the `catupgrd.sql` script instead of running `catctl.pl`), then you now must provide information for an additional input parameter, PARALLEL.

For example:

```
SQL> catupgrd.sql PARALLEL=NO
```

If you run `catupgrd.sql` without the parameter, then Oracle displays the following error message:

```
NOTE

The catupgrd.sql script is being deprecated in the 12.1 release of
Oracle Database. Customers are encouraged to use catctl.pl as
the replacement for catupgrd.sql when upgrading the database dictionary.

               cd $ORACLE_HOME/rdbms/admin
               $ORACLE_HOME/perl/bin/perl catctl.pl -n 4 catupgrd.sql

Refer to the Oracle Database Upgrade Guide for more information.

This database upgrade procedure must be called with the following
argument when invoking from the SQL prompt:

               @catupgrd.sql PARALLEL=NO
```

Changes for Deinstallation and Cleanup of Oracle Base

Review these topics to understand deinstallation and Oracle Base cleanup script changes.

- Change for Standalone Deinstallation Tool
  Starting with Oracle Database 12*c*, the deinstallation standalone utility is replaced with a deinstall option using Oracle Universal Installer (OUI).
- Desupport of CLEANUP_ORACLE_BASE Property
  In Oracle Database 12*c*, the `CLEANUP_ORACLE_BASE` property is removed for response file (silent) deinstalls.
- Deprecation of -cleanupOBase
  The `-cleanupOBase` flag of the deinstallation tool is deprecated in Oracle Database 12c Release 1 (12.1).

Change for Standalone Deinstallation Tool

Starting with Oracle Database 12*c*, the deinstallation standalone utility is replaced with a deinstall option using Oracle Universal Installer (OUI).

You can also run the deinstallation tool from the base directory of the installation media for Oracle Database, Oracle Database Client, or Oracle Grid Infrastructure.

Run the deinstallation tool by using the `runInstaller` command on Linux and UNIX, or `setup.exe` on Windows, with the `-deinstall` and `-home` options.

> **See Also:**
>
> - *Oracle Database Installation Guide* for your operating system for information about using OUI and `runInstaller`
> - *Oracle Grid Infrastructure Installation Guide* for your operating system

Desupport of CLEANUP_ORACLE_BASE Property

In Oracle Database 12*c*, the `CLEANUP_ORACLE_BASE` property is removed for response file (silent) deinstalls.

It is no longer supported to use CLEANUP_ORACLE_BASE to remove an Oracle base during silent or response file mode deinstalls.

Deprecation of -cleanupOBase

The `-cleanupOBase` flag of the deinstallation tool is deprecated in Oracle Database 12c Release 1 (12.1).

Identifying and Dropping Deprecated and Desupported Parameters

To locate deprecated parameters, you can use the `SELECT NAME FROM V$PARAMETER` query, and then drop those parameters with `ALTER SYSTEM RESET`.

Parameters can be deprecated or desupported after Oracle Database upgrades. Obsolete parameters generate ORA-32004 errors.

You can use the `ALTER SYSTEM RESET` command to drop the setting of any initialization parameter in the SPFILE that was used to start the instance. The `SCOPE = SPFILE` clause is not required, but you can include it.

Identifying Deprecated Parameters

To obtain a current list of deprecated parameters, run the following query in SQL*Plus:

```
SQL> SELECT name from v$parameter
         WHERE isdeprecated = 'TRUE' ORDER BY name;
```

The query returns a list of the deprecated parameters as of the date and time that you run it.

Dropping Deprecated Parameters Example

This SQL*Plus command drops the deprecated parameter `O7_DICTIONARY_ACCESSIBILITY` from all instances on a server with an SPFILE on the next system startup:

```
SQL> ALTER SYSTEM RESET O7_DICTIONARY_ACCESSIBILITY SCOPE=SPFILE SID='*'
```

- SEC_CASE_SENSITIVE_LOGON Initialization Parameter Deprecated
  The `SEC_CASE_SENSITIVE_LOGON` initialization parameter is deprecated in Oracle Database 12c release 1 (12.1).
- FILE_MAPPING Initialization Parameter Deprecated
  The `FILE_MAPPING` initialization parameter is deprecated. It is still supported for backward compatibility.
- RDBMS_SERVER_DN Initialization Parameter Deprecated
  The initialization parameter `RDBMS_SERVER_DN` is deprecated in Oracle Database release 12.1.0.2.

SEC_CASE_SENSITIVE_LOGON Initialization Parameter Deprecated

The `SEC_CASE_SENSITIVE_LOGON` initialization parameter is deprecated in Oracle Database 12c release 1 (12.1).

> **See Also:**
>
> *Oracle Database Reference* for information about the `SEC_CASE_SENSITIVE_LOGON` initialization parameter

FILE_MAPPING Initialization Parameter Deprecated

The `FILE_MAPPING` initialization parameter is deprecated. It is still supported for backward compatibility.

> **See Also:**
>
> *Oracle Database Reference* for information about the `FILE_MAPPING` initialization parameter

RDBMS_SERVER_DN Initialization Parameter Deprecated

The initialization parameter `RDBMS_SERVER_DN` is deprecated in Oracle Database release 12.1.0.2.

Use `LDAP_DIRECTORY_ACCESS` instead of `RDBMS_SERVER_DN`.

> **See Also:**
>
> *Oracle Database Reference* for information about this parameter

Changes to Security Auditing Features

The full set of auditing features are available automatically in Oracle Database 12c release 1 (12.1) and later releases.

The auditing functionality is redesigned in Oracle Database 12c. When you create a new database with Oracle Database 12c, the full set of auditing enhancement features are automatically available. If you upgrade from an earlier release, then you are given the option of using some of the new audit features and the audit functionality from the release from which you upgraded. Oracle strongly recommends that you migrate to the full set of the latest audit features.

> **See Also:**
>
> *Oracle Database Security Guide* for information about new auditing features and changes for security

Upgrading a System that Did Not Have SQLNET.ALLOWED_LOGON_VERSION Parameter Setting

Review the parameter setting for `SQLNET.ALLOWED_LOGON_VERSION_SERVER` to determine its implications for security and client connections to the upgraded database.

If you are upgrading a system that did not have a `SQLNET.ALLOWED_LOGON_VERSION` parameter setting (that is, it was using the default SQLNet setting `8`), then the new default value for the login verifier is set to `11`, which sets the 11g password verifier (`11G`) in Oracle Database 18c. This value permits Oracle Database 10g, 11g, 12c, and 18c clients to connect to the database, as well as Oracle Database 19c clients.

If you do not provide a parameter setting for `SQLNET.ALLOWED_LOGON_VERSION_SERVER` (or the deprecated `SQLNET.ALLOWED_LOGON_VERSION`) in the upgraded Oracle Database server, then the Oracle Database 19c default is `11`. This value enables connections from clients using releases earlier than Oracle Database release 11.2.0.3 that have not applied *critical patch update CPU Oct 2012*, or later patches, and that must use the Oracle Database 10g verifier (`10G`) to connect.

The higher the setting, the more restrictive the use of verifiers. A setting of `8` permits the most verifiers. For example, the `10G`, `11G`, and `12C` verifiers are all permitted with this setting. A setting of `12a` only permits the 12C verifier. For greater security, consider setting `SQLNET.ALLOWED_LOGON_VERSION_SERVER` to 12a. A setting of 12 permits both the 11G and 12C verifier to be used for authentication.

> **See Also:**
>
> *Oracle Database Security Guide* for additional information about verifier settings and client access

Oracle Warehouse Builder (OWB) Not Installed with Oracle Database

Oracle Warehouse Builder must be installed separately.

Starting with Oracle Database 12*c*, Oracle Warehouse Builder (OWB) is not installed as part of the software for Oracle Database. An installer for Oracle Warehouse Builder is available on Oracle Technology Network. OWB components that may exist from earlier releases are not upgraded as part of the Oracle Database upgrade process.

About Upgrading Oracle Database Release 10.2 or 11.1 and OCFS and RAW Devices

If you are upgrading an Oracle Database release 10.2.0.5 or release 11.1.0.7 environment that stores Oracle Clusterware files on OCFS on Windows or RAW devices, then you cannot directly upgrade to Oracle Database 12*c* release 1 (12.1).

You must first perform an interim upgrade to Oracle Database release 11.2, and then migrate Oracle Clusterware files to Oracle Automatic Storage Management (Oracle ASM). Then you can upgrade from release 11.2 to Oracle Database 12*c* release 1 (12.1).

> **See Also:**
>
> - http://www.oracle.com/technetwork/documentation/index.html to obtain *Oracle Database Upgrade Guide* for Oracle Database 11*g* Release 2 (11.2)
> - *Oracle Database Installation Guide* for your operating system for procedures on installing the Oracle Database software
> - *Oracle Automatic Storage Management Administrator's Guide* for information about migrating a database to use Oracle Automatic Storage Management (Oracle ASM)

Change to VARCHAR2, NVARCHAR2, and RAW Datatypes

You can increase the MAX_STRING_SIZE value for these datatypes to `32767` bytes in Oracle Database 12c Release 1 (12.1) and later releases.

Starting with Oracle Database 12c Release 1 (12.1), you can increase the maximum size of the VARCHAR2, NVARCHAR2, and RAW datatypes to `32767` bytes. This size increase is possible if you set the `COMPATIBLE` initialization parameter to `12.0` or higher, and you set the `MAX_STRING_SIZE` initialization parameter to `EXTENDED`. This

size increase is available for non-CDB Oracle Database instances, and for PDBs in multitenant architecture.

If you want to make this change, then you must run the `utl32k.sql` script in `UPGRADE` mode. After you run the script, you can set `MAX_STRING_SIZE` to `EXTENDED`.

> **Caution:**
>
> When you increase the COMPATIBLE initialization to 12.0 or higher, you cannot reverse this change.

Changed Default for RESOURCE_LIMIT Parameter

RESOURCE_LIMIT is set to TRUE by default.

The Oracle Database `RESOURCE_LIMIT` parameter determines if resource limits are enforced in database profiles. In this release, the `RESOURCE_LIMIT` parameter is set to `TRUE` by default. If Oracle resource limits are disabled, then any defined profile limits are ignored. This behavior does not apply to password resources.

Check the resource_limit parameter setting by entering the following SQL*Plus command:

```
SQL> select value from v$parameter where name = 'resource_limit';
```

Oracle XML DB is Mandatory and Cannot Be Uninstalled

Starting with Oracle Database 12c, Oracle XML DB is a mandatory component of Oracle Database.

You cannot uninstall Oracle XML DB, and there is no option to exclude it when you create an Oracle Database. Oracle XML DB is automatically installed or upgraded when you upgrade an existing Oracle Database to Oracle Database 12c.

Direct NFS Enabled By Default for Oracle RAC

In Oracle Database 12c release 1 (12.1) and later releases, Direct NFS (DNFS) is enabled by default with Oracle Real Application Clusters (Oracle RAC) installations.

By default, Direct NFS Client is installed in an enabled state for Oracle RAC installations. This is different from Oracle Database single-instance installations, in which you have to enable Direct NFS, and the Direct NFS Client is disabled by default as part of installation. If you upgrade an Oracle RAC Oracle Database release to Release 12.1 or later releases, and you do not have Direct NFS enabled on your source database, then you encounter the error message `ORA-17500: ODM err:Operation not permitted`.

For more information about this change, and to review options in preparation for your upgrade, refer to My Oracle Support notes 954425.1, and 1966267.1

Related Topics

- My Oracle Support Doc ID 954425.1

- My Oracle Support Doc ID 1966267.1

Deprecated Features in Oracle Database 12c Release 1 (12.1)

Review the deprecated features listed in this section to prepare to use alternatives after you upgrade. Features deprecated in an earlier release can be desupported in a later release.

- Deprecation of Non-CDB Architecture
  The non-CDB architecture is deprecated in Oracle Database 12*c*. It can be desupported and unavailable in a release after Oracle Database 19c.
- Deprecation of catupgrd.sql Script and Introduction of Parallel Upgrade Utility
  In Oracle Database 12c release 1 (12.1), Oracle recommends that you use the Parallel Upgrade Utility (`catctl.plinstead` of `catupgrd.sql` to enable parallel upgrades.
- DELETE_CATALOG_ROLE Deprecated
  The `DELETE_CATALOG_ROLE` database role is deprecated in Oracle Database 12*c*.
- Deprecated Functions and Parameters in Oracle Label Security
  Nine Oracle Label Security features are deprecated in Oracle Database 12*c* release 1.
- Deprecation of IGNORECASE and SEC_CASE_SENSITIVE_LOGON
  The `IGNORECASE` argument of `ORAPWD` and the `SEC_CASE_SENSITIVE_LOGON` system parameter are deprecated in Oracle Database 12*c*.
- Deprecation of SQLNET.ALLOWED_LOGON_VERSION Parameter
  The `SQLNET.ALLOWED_LOGON_VERSION` parameter is deprecated in Oracle Database 12*c*.
- Deprecation of Windows NTS Authentication Using the NTLM Protocol
  Because of security vulnerabilities, NTLM is deprecated as of Oracle Database 12*c*.
- Deprecation of Public Key Infrastructure for Transparent Data Encryption
  Public Key Infrastructure (PKI) is deprecated for Transparent Data Encryption (TDE) in Oracle Database 12*c*.
- Oracle Data Guard Broker Deprecated Features
  Review these deprecations if you use Oracle Data Guard Broker.
- Oracle Data Pump Export Utility Features
  The `XML_CLOBS` option of the Oracle Data Pump Export `DATA_OPTIONS` parameter is deprecated.
- Deprecated Default Rule Sets for Oracle Database Vault
  Rule sets listed here are deprecated with Oracle Database 12c Release 1.
- Deprecated Default Realms for Oracle Database Vault
  The Oracle Data Dictionary realm and Oracle Enterprise Manager realm are deprecated in this release.
- Deprecated API for Oracle Database Vault
  The `DVSYS.DBMS_MACADM.SYNC_RULES` procedure is deprecated, because its functionality is built into the rule creation functionality.
- VPD Support in Oracle Database Semantic Technologies is Deprecated
  The Virtual Private Database (VPD) is deprecated for Oracle Database Semantic Technologies in Oracle Database 12*c* (12.1).

- Version-Enabled Models Support In Oracle Database Semantic Technologies
  The version-enabled models feature is deprecated for Oracle Database Semantic Technologies in Oracle Database 12*c* (12.1).
- Deprecation of Oracle Streams
  Oracle Streams is deprecated in Oracle Database 12*c*. It can be desupported and unavailable in a later Oracle Database release.
- Deprecation of Advanced Replication
  Oracle Database Advanced Replication is deprecated in Oracle Database 12c.
- Deprecation of Single-Character SRVCTL CLI Options
  Starting with Oracle Database 12*c*, single-character options are deprecated. They can be desupported in a later release.
- Deprecation of Stored List of Administrative Users for Cluster Administration
  Cluster administration is managed differently starting with Oracle Database 12*c*.
- Deprecation of -checkpasswd for QOSCTL Quality of Service (QoS) Command
  The syntax `qosctl -checkpasswd username password` is deprecated.
- Deprecation of SQLJ Inside the Server
  SQLJ usage inside the database server is deprecated in this release.
- Deprecated Oracle Update Batching
  Oracle Update Batching APIs are deprecated in Oracle Database 12c.
- Deprecated EndToEndMetrics-related APIs
  `EndToEndMetrics` -related APIs are deprecated in Oracle Database 12*c*. Use Universal Connection Pool instead.
- Deprecated Stored Outlines
  Stored outlines are deprecated in Oracle Database 12*c*.
- Deprecated Concrete Classes in oracle.sql Package
  The concrete classes in the `oracle.sql` package are deprecated in Oracle Database 12*c*.
- Deprecated defineColumnType Method
  The JDBC method `defineColumnType` is deprecated in Oracle Database 12*c* Release 1.
- Deprecated CONNECTION_PROPERTY_STREAM_CHUNK_SIZE Property
  The JDBC property CONNECTION_PROPERTY_STREAM_CHUNK_SIZE is deprecated in this release.
- Deprecated Features for Oracle Call Interface
  Oracle Call Interface (OCI) features listed here are deprecated in Oracle Database 12*c*. They can be desupported in a future release
- DICOM in Oracle Multimedia Deprecated in Oracle Database 12c Release 1 (12.1)
  In Oracle Database 12c, ORDImage support for Oracle Multimedia DICOM is deprecated.
- Deprecated NT LAN Manager (NTLM) Protocol for Oracle Net Services
  The NT LAN Manager (NTLM) protocol for domain authentication is deprecated in the Oracle Windows adapter.

Deprecation of Non-CDB Architecture

The non-CDB architecture is deprecated in Oracle Database 12*c*. It can be desupported and unavailable in a release after Oracle Database 19c.

Oracle recommends use of the CDB architecture.

> **Note:**
>
> There remain a small number of features that do not work with the CDB architecture (see README, section 2.2.1 "Features Restricted or Not Available for a Multitenant Container Database"). If you need these features, then continue to use the non-CDB architecture until your required feature works with the CDB architecture.

Deprecation of catupgrd.sql Script and Introduction of Parallel Upgrade Utility

In Oracle Database 12c release 1 (12.1), Oracle recommends that you use the Parallel Upgrade Utility (`catctl.pl`instead of `catupgrd.sql` to enable parallel upgrades.

Oracle Database 12c release 1 (12.1) introduces the new Parallel Upgrade Utility, `catctl.pl`. This utility replaces the `catupgrd.sql` script that was used in earlier releases. Although you can still use the `catupgrd.sql` script, it is deprecated starting with Oracle Database 12c. It will be removed in future releases. Oracle recommends that you perform database upgrades by using the new Parallel Upgrade Utility.

DELETE_CATALOG_ROLE Deprecated

The `DELETE_CATALOG_ROLE` database role is deprecated in Oracle Database 12*c*.

> **See Also:**
>
> *Oracle Database Reference* for information about this role and dictionary objects

Deprecated Functions and Parameters in Oracle Label Security

Nine Oracle Label Security features are deprecated in Oracle Database 12*c* release 1.

The Oracle Label Security features listed here are deprecated in Oracle Database 12*c* release 1. They can be desupported in a future release. Oracle recommends that you use the alternative features listed here.

- LEAST_UBOUND. Use OLS_GREATEST_LBOUND instead.
- LUBD. Use OLS_GLBD instead.
- DOMINATES. Use OLS_DOMINATES instead.
- DOM. Use OLS_STRICTLY_DOMINATES instead.
- STRICTLY_DOMINATES. Use OLS_STRICTLY_DOMINATES instead.
- S_DOM. Use OLS_STRICTLY_DOMINATES instead.
- DOMINATED_BY. Use OLS_DOMINATED_BY instead.
- DOM_BY. Use OLS_DOMINATED_BY instead.

- STRICTLY_DOMINATED_BY. Use OLS_STRICTLY_DOMINATED_BY instead.
- S_DOM_BY. Use OLS_STRICTLY_DOMINATED_BY instead.

See Also:

- *Oracle Label Security Administrator's Guide* for information about finding greatest lower bound with GREATEST_LBOUND
- *Oracle Label Security Administrator's Guide* for information about OLS_GLBD
- *Oracle Label Security Administrator's Guide* for information about the OLS_DOMINATES standalone function
- *Oracle Label Security Administrator's Guide* for information about OLS_STRICTLY_DOMINATES standalone function
- *Oracle Label Security Administrator's Guide* for information about OLS_DOMINATED_BY standalone function
- *Oracle Label Security Administrator's Guide* for information about OLS_STRICTLY_DOMINATED_BY standalone function

Deprecation of IGNORECASE and SEC_CASE_SENSITIVE_LOGON

The `IGNORECASE` argument of `ORAPWD` and the `SEC_CASE_SENSITIVE_LOGON` system parameter are deprecated in Oracle Database 12*c*.

By default, passwords in Oracle Database 12*c* are case-sensitive.

See Also:

Oracle Database Security Guide for more information about configuring authentication and password protection

Deprecation of SQLNET.ALLOWED_LOGON_VERSION Parameter

The `SQLNET.ALLOWED_LOGON_VERSION` parameter is deprecated in Oracle Database 12*c*.

`SQLNET.ALLOWED_LOGON_VERSION` is replaced with two new Oracle Net Services parameters:

- `SQLNET.ALLOWED_LOGON_VERSION_SERVER`
- `SQLNET.ALLOWED_LOGON_VERSION_CLIENT`

Related Topics

- Upgrading a System that Did Not Have SQLNET.ALLOWED_LOGON_VERSION Parameter Setting
  Review the parameter setting for `SQLNET.ALLOWED_LOGON_VERSION_SERVER` to determine its implications for security and client connections to the upgraded database.
- Check for the SQLNET.ALLOWED_LOGON_VERSION Parameter Behavior
  Connections to Oracle Database from clients earlier than release 10g fail with the error `ORA-28040: No matching authentication protocol`.

> **See Also:**
>
> *Oracle Database Net Services Reference* for information about `SQLNET.ALLOWED_LOGON_VERSION_SERVER`
>
> *Oracle Database Net Services Reference* for information about `SQLNET.ALLOWED_LOGON_VERSION_CLIENT`
>
> *Oracle Database Security Guide* for information about this deprecation

Deprecation of Windows NTS Authentication Using the NTLM Protocol

Because of security vulnerabilities, NTLM is deprecated as of Oracle Database 12*c*.

Windows users can no longer authenticate using the NTS adaptor on Windows clients and servers that require the NT Lan Manager (NTLM) protocol. Windows users can still use Kerberos. NTLM is still used for local user authentication, and in cases in which the database service runs as a local user.

A new client side `sqlnet.ora` boolean parameter NO_NTLM (defaulting to false) allows you to control when NTLM is used in NTS authentication. When you set NO_NTLM to true, this parameter value prevents NTLM from being used in Windows NTS authentication.

> **See Also:**
>
> *Oracle Database Platform Guide for Microsoft Windows* for information on changes that affect the Windows platform in this release

Deprecation of Public Key Infrastructure for Transparent Data Encryption

Public Key Infrastructure (PKI) is deprecated for Transparent Data Encryption (TDE) in Oracle Database 12*c*.

To configure TDE, use the ADMINISTER KEY MANAGEMENT SQL statement. Other implementations of PKI are not affected.

> **See Also:**
>
> *Oracle Database Advanced Security Administrator's Guide* for information about configuring TDE

Oracle Data Guard Broker Deprecated Features

Review these deprecations if you use Oracle Data Guard Broker.

As part of Oracle Data Guard Broker's support for Separation of Duty features, the following broker properties are deprecated in Oracle Database 12*c* :

- `LsbyASkipCfgPr`
- `LsbyASkipErrorCfgPr`
- `LsbyASkipTxnCfgPr`
- `LsbyDSkipCfgPr`
- `LsbyDSkipErrorCfgPr`
- `LsbyDSkipTxnCfgPr`
- `LsbySkipTable`
- `LsbySkipTxnTable`

There are no replacements.

Oracle Data Pump Export Utility Features

The `XML_CLOBS` option of the Oracle Data Pump Export `DATA_OPTIONS` parameter is deprecated.

> **See Also:**
>
> *Oracle Database Utilities* for information about the Export `DATA_OPTIONS` parameter

Deprecated Default Rule Sets for Oracle Database Vault

Rule sets listed here are deprecated with Oracle Database 12c Release 1.

The following rule sets are deprecated in this release:

- Allow Oracle Data Pump Operation rule set
- Allow Scheduler Job rule set

Related Topics

- *Oracle Database Vault Administrator's Guide*

Deprecated Default Realms for Oracle Database Vault

The Oracle Data Dictionary realm and Oracle Enterprise Manager realm are deprecated in this release.

The objects formerly protected by the Oracle Data Dictionary realm have been migrated to new realms.

> **See Also:**
>
> *Oracle Database Vault Administrator's Guide* for information about new realms and default realms

Deprecated API for Oracle Database Vault

The `DVSYS.DBMS_MACADM.SYNC_RULES` procedure is deprecated, because its functionality is built into the rule creation functionality.

> **See Also:**
>
> *Oracle Database Vault Administrator's Guide* for information on rule creation

VPD Support in Oracle Database Semantic Technologies is Deprecated

The Virtual Private Database (VPD) is deprecated for Oracle Database Semantic Technologies in Oracle Database 12*c* (12.1).

Transition existing Semantic Technologies applications that depend on Virtual Private Database (VPD) to use Oracle Label Security (OLS) instead.

> **See Also:**
>
> *Oracle Database Semantic Technologies Developer's Guide* for information about fine-grained access control for RDF data

Version-Enabled Models Support In Oracle Database Semantic Technologies

The version-enabled models feature is deprecated for Oracle Database Semantic Technologies in Oracle Database 12*c* (12.1).

The specific alternative to using Workspace Manager with semantic data depends on the purpose of the application.

> **See Also:**
>
> *Oracle Database Semantic Technologies Developer's Guide* for information about Workspace Manager support for RDF data
>
> https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1468273.1 for more information about use cases and recommended alternatives

Deprecation of Oracle Streams

Oracle Streams is deprecated in Oracle Database 12c. It can be desupported and unavailable in a later Oracle Database release.

Use Oracle GoldenGate to replace all replication features of Oracle Streams.

> **Note:**
>
> Oracle Database Advanced Queuing is independent of Oracle Streams, and continues to be enhanced.

> **See Also:**
>
> Oracle GoldenGate documentation for information

Deprecation of Advanced Replication

Oracle Database Advanced Replication is deprecated in Oracle Database 12c.

Read-only materialized views are still supported with basic replication.

Use Oracle GoldenGate to replace all features of Advanced Replication. This guidance includes multimaster replication, updateable materialized views, and deployment templates.

> **See Also:**
>
> *Oracle GoldenGate documentation*

Deprecation of Single-Character SRVCTL CLI Options

Starting with Oracle Database 12*c*, single-character options are deprecated. They can be desupported in a later release.

The Server Control Utility (SRVCTL) command line interface (CLI) supports long GNU-style options in addition to short CLI options used in earlier releases.

> **See Also:**
>
> *Oracle Real Application Clusters Administration and Deployment Guide* for information about SRVCTL

Deprecation of Stored List of Administrative Users for Cluster Administration

Cluster administration is managed differently starting with Oracle Database 12*c*.

Starting with Oracle Database 12*c*, the method of cluster administration using a stored list of administrative users is being replaced with more comprehensive management of administrative user roles by configuring the access control list of the policy set.

> **See Also:**
>
> *Oracle Clusterware Administration and Deployment Guide* for more information about cluster administration

Deprecation of -checkpasswd for QOSCTL Quality of Service (QoS) Command

The syntax `qosctl -checkpasswd username password` is deprecated.

> **See Also:**
>
> *Oracle Database Quality of Service Management User's Guide* for information about QOSCTL syntax and commands

Deprecation of SQLJ Inside the Server

SQLJ usage inside the database server is deprecated in this release.

The capability of translating and running SQLJ applications inside the database will not be available in later releases. SQLJ can only be used as a client tool to translate the applications that can connect to Oracle Database and run as a client. SQLJ cannot be used inside stored procedures, functions, or triggers.

Deprecated Oracle Update Batching

Oracle Update Batching APIs are deprecated in Oracle Database 12c.

The following APIs are deprecated and marked deprecated in the JDBC Javadoc:

- `OraclePreparedStatement.setExecuteBatch()`
- `OraclePreparedStatement.getExecuteBatch()`
- `OracleCallableStatement.setExecuteBatch()`

Use Standard Update Batching instead.

Related Topics

- *Oracle Database JDBC Developer's Guide*

Deprecated EndToEndMetrics-related APIs

`EndToEndMetrics` -related APIs are deprecated in Oracle Database 12*c*. Use Universal Connection Pool instead.

The following APIs are deprecated and marked deprecated in the JDBC Javadoc:

- `getEndToEndMetrics`
- `getEndToEndECIDSequenceNumber`
- `setEndToEndMetrics`
- `setApplicationContext`
- `clearAllApplicationContext`

> **See Also:**
>
> *Oracle Database JDBC Developer's Guide* and the JDBC Javadoc for information about Universal Connection Pool

Deprecated Stored Outlines

Stored outlines are deprecated in Oracle Database 12*c*.

Use plan baselines instead.

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* for information about migrating stored outlines to plan baselines

Deprecated Concrete Classes in oracle.sql Package

The concrete classes in the `oracle.sql` package are deprecated in Oracle Database 12*c*.

These classes are replaced with new interfaces in the `oracle.jdbc` package.

> **See Also:**
>
> *Oracle Database JDBC Developer's Guide* for information about the new interfaces for `oracle.jdbc`

Deprecated defineColumnType Method

The JDBC method `defineColumnType` is deprecated in Oracle Database 12*c* Release 1.

Deprecated CONNECTION_PROPERTY_STREAM_CHUNK_SIZE Property

The JDBC property CONNECTION_PROPERTY_STREAM_CHUNK_SIZE is deprecated in this release.

Deprecated Features for Oracle Call Interface

Oracle Call Interface (OCI) features listed here are deprecated in Oracle Database 12*c*. They can be desupported in a future release

- OCI deployment parameters in sqlnet.ora are deprecated. These include the following parameters:
  - OCI client result cache parameters: `OCI_RESULT_CACHE_MAX_SIZE`, `OCI_RESULT_CACHE_MAX_RSET_SIZE`, and `OCI_RESULT_CACHE_MAX_RSET_ROWS`

DICOM in Oracle Multimedia Deprecated in Oracle Database 12c Release 1 (12.1)

In Oracle Database 12c, ORDImage support for Oracle Multimedia DICOM is deprecated.

Deprecated NT LAN Manager (NTLM) Protocol for Oracle Net Services

The NT LAN Manager (NTLM) protocol for domain authentication is deprecated in the Oracle Windows adapter.

Only Kerberos authentication is used for the NTS adapter.

> **See Also:**
>
> *Oracle Database Net Services Reference* for information on Kerberos parameters for the `sqlnet.ora` file

Deprecated Views in Oracle Database 12c Release 1 (12.1)

Review the deprecated features listed in this section to prepare to use alternatives after you upgrade. Features deprecated in an earlier release can be desupported in a later release.

The following views are deprecated:

- `ALL_SCHEDULER_CREDENTIALS` view. See *Oracle Database Reference* for more information.
- `DBA_NETWORK_ACL_PRIVILEGES` view. See *Oracle Database Reference* for more information.
- `DBA_NETWORK_ACLS` view. See *Oracle Database Reference* for more information.
- `DBA_SCHEDULER_CREDENTIALS` view. See *Oracle Database Reference* for more information.
- `USER_NETWORK_ACL_PRIVILEGES` view. See *Oracle Database Reference* for more information.
- `USER_SCHEDULER_CREDENTIALS` view. See *Oracle Database Reference* for more information.
- `V$OBJECT_USAGE` view. Use the `USER_OBJECT_USAGE` view instead. See *Oracle Database Reference* for more information.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for information about specifying Scheduler job credentials

Deprecated Initialization Parameters in Oracle Database 12c Release 1 (12.1)

To understand changes and replacements in parameter settings, review the parameters deprecated in the 12.1 release. These parameters can be removed in a later release.

- Identifying and Dropping Deprecated and Desupported Parameters
  To locate deprecated parameters, you can use the `SELECT NAME FROM V$PARAMETER` query, and then drop those parameters with `ALTER SYSTEM RESET`.

- FILE_MAPPING Initialization Parameter Deprecated
  The FILE_MAPPING initialization parameter is deprecated. It is still supported for backward compatibility.
- RDBMS_SERVER_DN Initialization Parameter Deprecated
  The initialization parameter RDBMS_SERVER_DN is deprecated in Oracle Database release 12.1.0.2.
- SEC_CASE_SENSITIVE_LOGON Initialization Parameter Deprecated
  The SEC_CASE_SENSITIVE_LOGON initialization parameter is deprecated in Oracle Database 12c release 1 (12.1).

Identifying and Dropping Deprecated and Desupported Parameters

To locate deprecated parameters, you can use the SELECT NAME FROM V$PARAMETER query, and then drop those parameters with ALTER SYSTEM RESET.

Parameters can be deprecated or desupported after Oracle Database upgrades. Obsolete parameters generate ORA-32004 errors.

You can use the ALTER SYSTEM RESET command to drop the setting of any initialization parameter in the SPFILE that was used to start the instance. The SCOPE = SPFILE clause is not required, but you can include it.

Identifying Deprecated Parameters

To obtain a current list of deprecated parameters, run the following query in SQL*Plus:

```
SQL> SELECT name from v$parameter
         WHERE isdeprecated = 'TRUE' ORDER BY name;
```

The query returns a list of the deprecated parameters as of the date and time that you run it.

Dropping Deprecated Parameters Example

This SQL*Plus command drops the deprecated parameter O7_DICTIONARY_ACCESSIBILITY from all instances on a server with an SPFILE on the next system startup:

```
SQL> ALTER SYSTEM RESET O7_DICTIONARY_ACCESSIBILITY SCOPE=SPFILE SID='*'
```

FILE_MAPPING Initialization Parameter Deprecated

The FILE_MAPPING initialization parameter is deprecated. It is still supported for backward compatibility.

> **See Also:**
>
> *Oracle Database Reference* for information about the FILE_MAPPING initialization parameter

RDBMS_SERVER_DN Initialization Parameter Deprecated

The initialization parameter `RDBMS_SERVER_DN` is deprecated in Oracle Database release 12.1.0.2.

Use `LDAP_DIRECTORY_ACCESS` instead of `RDBMS_SERVER_DN`.

> **See Also:**
>
> *Oracle Database Reference* for information about this parameter

SEC_CASE_SENSITIVE_LOGON Initialization Parameter Deprecated

The `SEC_CASE_SENSITIVE_LOGON` initialization parameter is deprecated in Oracle Database 12c release 1 (12.1).

> **See Also:**
>
> *Oracle Database Reference* for information about the `SEC_CASE_SENSITIVE_LOGON` initialization parameter

Desupported Features in Oracle Database 12c Release 1(12.1)

Review this list of desupported features as part of your upgrade planning.

- Desupport of Oracle Enterprise Manager Database Control
  Starting with Oracle Database 12*c*, Oracle Enterprise Manager Database Control is desupported and is no longer available.
- Desupported Cipher Suites for Secure Sockets Layer (SSL)
  Review this list of desupported cipher suites if you use Oracle Advanced Security.
- Desupport of Database Rules Manager (RUL) and Expression Filter (EXF)
  Starting with Oracle Database 12*c* release 1, the Expression Filter (EXF) and Database Rules Manager (RUL) features are desupported.
- Desupport of Oracle COM Automation on Windows
  Oracle Database 12*c* does not contain Oracle COM Automation.
- Desupport of Oracle Objects for OLE
  Oracle Database 12*c* does not contain Oracle Objects for OLE on Microsoft Windows systems.
- Desupport of Oracle Counters for Windows Performance Monitor
  Oracle Database 12*c* does not contain Oracle Counters for Windows Performance Monitor.

- Programming Interface Deprecations for Oracle Data Provider for .NET
  Oracle Data Provider for .NET application programming interfaces for Transaction Guard listed here are deprecated in Oracle Database 12*c* (12.1.0.2). They can be desupported in a future release.
- Desupport of Oracle Cluster File System (OCFS) on Windows
  Starting with Oracle Database 12*c*, Oracle Cluster File System (OCFS) is desupported on Windows. Support and distribution of OCFS on Linux (OCFS and OCFS2) remains unaffected by this desupport notice.
- Desupport for Raw Storage Devices
  Starting with Oracle Database 12*c* release 1 (12.1), block file storage on raw devices is not supported.
- Desupport of cluvfy comp cfs for OCFS
  The `cluvfy comp cfs` component verification command option is removed from Oracle Database 12c release 1 (12.1).
- Desupport of Change Data Capture
  Oracle Change Data Capture is not included in Oracle Database 12*c* and is replaced with Oracle GoldenGate.
- Desupported Features in Oracle Data Mining
  These Oracle Data Mining features are desupported in Oracle Database 12*c*.
- Desupported Implicit Connection Caching
  Implicit Connection Caching is desupported in Oracle Database 12*c*.
- Desupport of ABN Models for Oracle Data Mining Upgrades
  Starting with Oracle Database 12c, Oracle is desupporting the Data Mining Java API and the Adaptive Bayes Network (ABN) algorithm.
- Desupport of OLAP Catalog (AMD)
  Starting with Oracle Database 12c, the Common Warehouse Metamodel (CWM) standard is desupported for the OLAP catalog (AMD).
- Desupport of CSSCAN and CSALTER for Oracle Globalization
  Oracle Database 12*c* includes Oracle Database Migration Assistant for Unicode (DMU), and Oracle is desupporting the legacy database tools CSSCAN and CSALTER.
- Desupport of Oracle Net Connection Pooling
  In Oracle Database 12*c*, Oracle Net connection pooling is no longer supported.
- Desupport of Oracle Names
  Oracle Names was desupported as a naming method in Oracle Database 11*g*. You must migrate to directory naming.
- Desupport of Oracle Net Listener Password
  In Oracle Database 12c, the Oracle Net Listener password feature is no longer supported.
- Desupport of SQLNET.KERBEROS5_CONF_MIT Parameter for Oracle Net Services
  The `SQLNET.KERBEROS5_CONF_MIT` networking parameter is no longer needed and is not supported in `sqlnet.ora`.
- Desupport of Oracle Names Control Utility for Oracle Net Services
  The Oracle Names Control Utility is desupported. Starting with Oracle Database 10*g*, it is not available.

- Desupport of CTXXPATH in Oracle Text and Oracle XML DB
  The `CTXSYS.CTXXPATH` index is desupported, starting with Oracle Database 12c Release 1 (12.1).
- Desupport of ALTER INDEX OPTIMIZE for Text Indexes
  The ALTER INDEX OPTIMIZE `[token index_token | fast | full [maxtime (time | unlimited)]` operation is not supported for Oracle Database 12*c*.
- Desupport of SYNC [MEMORY memsize] for Text Indexes
  The `SYNC [MEMORY memsize]` operation is not supported for Oracle Database 12c

Desupport of Oracle Enterprise Manager Database Control

Starting with Oracle Database 12*c*, Oracle Enterprise Manager Database Control is desupported and is no longer available.

Oracle introduces Oracle Enterprise Manager Database Express (Oracle EM Express) as a replacement. Oracle EM Express is installed when you upgrade to Oracle Database 12*c*.

You can carry out a manual configuration of the HTTP port for Oracle EM Express:

1. Look in the `init.ora/spfile` (default setting) for the following string:

   ```
   dispatchers=(PROTOCOL=TCP)(SERVICE=sample XDB)
   ```

   Check the Oracle EM Express port configuration:

   ```
   SQL> select DBMS_XDB_CONFIG.getHTTPport() from dual;
   SQL> select DBMS_XDB_CONFIG.getHTTPSport() from dual;
   ```

2. Set new ports. For example:

   ```
   SQL> exec DBMS_XDB_CONFIG.setHTTPport(5500);
   SQL> exec DBMS_XDB_CONFIG.setHTTPSport(8080);
   ```

3. Access the Oracle EM Express home page in the browser. For example, using the settings you have provided, check the following path, where *`database`* is your SID, *`hostname`* is your fully qualified domain name, and *`port`* is the port you assigned:

   `http://database-hostname:port/em`

   For example:

   ```
   http://localhost:5500/em
   ```

4. Repeat this configuration step for the CDB, and for every PDB, using different port numbers.

> **See Also:**
>
> - *Oracle Database Concepts* for an overview of Oracle EM Express
> - *Oracle Database 2 Day DBA* for information about using Oracle EM Express
> - *Oracle Database Concepts* for an introduction to Oracle Enterprise Manager Cloud Control

Desupported Cipher Suites for Secure Sockets Layer (SSL)

Review this list of desupported cipher suites if you use Oracle Advanced Security.

Oracle Advanced Security has desupported the following cipher suites in Oracle Database 12*c*:

- `SSL_DH_anon_WITH_DES_CBC_SHA`
- `SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_RSA_EXPORT_WITH_RC4_40_MD5`
- `SSL_RSA_WITH_DES_CBC_SHA`

> **See Also:**
>
> *Oracle Database Security Guide* for information about supported SSL cipher suites

Desupport of Database Rules Manager (RUL) and Expression Filter (EXF)

Starting with Oracle Database 12*c* release 1, the Expression Filter (EXF) and Database Rules Manager (RUL) features are desupported.

If you are using Rules Manager, then Oracle recommends that you migrate to Oracle Business Rules, which is a component of Oracle Fusion Middleware. The Continuous Query Notification feature of Oracle Database replaces Expression Filter.

This script is executed by the upgrade process. To remove these components before upgrading, run the `catnoexf.sql` script before the upgrade. The `catnoexf.sql` script is located under *`ORACLE_HOME`*`/rdbms/admin/`.

> **See Also:**
>
> https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1244535.1

Desupport of Oracle COM Automation on Windows

Oracle Database 12*c* does not contain Oracle COM Automation.

This feature was deprecated in Oracle Database 11*g*, which is the last database release that contains the database component Oracle COM Automation. Oracle recommends that you migrate your Oracle COM applications to current technology, such as the .NET Framework.

See Also:

https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1175293.1

Desupport of Oracle Objects for OLE

Oracle Database 12*c* does not contain Oracle Objects for OLE on Microsoft Windows systems.

Oracle Objects for OLE is deprecated in Oracle Database 11*g*. You can migrate your code to the OLE DB data access standard and ActiveX Data Objects (ADO), or you can migrate your applications to .NET (or Java or another application architecture) and use another driver.

See Also:

https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1175303.1

Desupport of Oracle Counters for Windows Performance Monitor

Oracle Database 12*c* does not contain Oracle Counters for Windows Performance Monitor.

Oracle Counters for Windows Performance Monitor was deprecated in Oracle Database 11*g*. The counters are not installed by default in earlier releases, and the counters only work on Windows. For monitoring, Oracle recommends that you use Oracle Enterprise Manager Cloud Control.

See Also:

https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1175297.1

Programming Interface Deprecations for Oracle Data Provider for .NET

Oracle Data Provider for .NET application programming interfaces for Transaction Guard listed here are deprecated in Oracle Database 12*c* (12.1.0.2). They can be desupported in a future release.

- `OracleLogicalTransactionStatus` class
- `OracleConnection.LogicalTransactionId` property
- `OracleConnection.GetLogicalTransactionStatus` method

Desupport of Oracle Cluster File System (OCFS) on Windows

Starting with Oracle Database 12*c*, Oracle Cluster File System (OCFS) is desupported on Windows. Support and distribution of OCFS on Linux (OCFS and OCFS2) remains unaffected by this desupport notice.

Databases currently using OCFS on Windows to host either the Oracle cluster files (Oracle Cluster Registry and voting files) or database files or both need to have these files migrated off OCFS *before* upgrading to Oracle Database 12*c*.

See Also:

- *Oracle Automatic Storage Management Administrator's Guide* for information about migrating a database to use Oracle Automatic Storage Management (Oracle ASM)
- https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1392280.1

Desupport for Raw Storage Devices

Starting with Oracle Database 12*c* release 1 (12.1), block file storage on raw devices is not supported.

You must migrate any data files stored on raw devices to Oracle ASM, to a cluster file system, or to a Network File System (NFS).

This desupport guideline also applies to OCR and voting disk files for Oracle Clusterware. You cannot store OCR or voting disk files on raw devices. Before you start the upgrade, you must move Oracle Clusterware files from raw devices to a supported storage option.

Desupport of cluvfy comp cfs for OCFS

The `cluvfy comp cfs` component verification command option is removed from Oracle Database 12c release 1 (12.1).

The Cluster Verification Utility (`cluvfy`) option `comp cfs` is removed in Oracle Database 12.1, because Oracle Cluster File System (OCFS) is no longer supported.

> **See Also:**
>
> *Oracle Clusterware Administration and Deployment Guide* for more information about cluster administration

Desupport of Change Data Capture

Oracle Change Data Capture is not included in Oracle Database 12*c* and is replaced with Oracle GoldenGate.

> **See Also:**
>
> *Oracle GoldenGate documentation* for information

Desupported Features in Oracle Data Mining

These Oracle Data Mining features are desupported in Oracle Database 12*c*.

- The Oracle Data Mining Java API is no longer available. The programmatic interfaces to Oracle Data Mining 12*c* consist of two PL/SQL packages, `DBMS_DATA_MINING` and `DBMS_DATA_MINING_TRANSFORM`, and a family of SQL language functions for scoring data.

  > **See Also:**
  >
  > *Oracle Data Mining User's Guide* for information about the Data Mining PL/SQL packages

- The Adaptive Bayes Network (ABN) algorithm is no longer available. The Decision Tree algorithm replaces ABN in Oracle Data Mining 12*c*.

  > **See Also:**
  >
  > *Oracle Data Mining Concepts* for information about the Decision Tree algorithm

Desupported Implicit Connection Caching

Implicit Connection Caching is desupported in Oracle Database 12*c*.

Use Universal Connection Pool instead.

> **See Also:**
>
> *Oracle Universal Connection Pool for JDBC Developer's Guide* for information

Desupport of ABN Models for Oracle Data Mining Upgrades

Starting with Oracle Database 12c, Oracle is desupporting the Data Mining Java API and the Adaptive Bayes Network (ABN) algorithm.

Because Oracle is desupporting these features, you cannot upgrade models created by the Oracle Data Mining Java API from Oracle Database release 11g to Oracle Database 12c. All other models and metadata are upgraded automatically during the upgrade from Oracle Database 11g to Oracle Database 12c. ABN models can be upgraded, but you cannot use them in an Oracle Database 12c database. You must drop ABN models either before the upgrade or afterward. You can replace ABN models by building new classification models in the Oracle Database 12c database.

> **See Also:**
>
> *Oracle Data Mining User's Guide* for information about the Data Mining API and data mining models

Desupport of OLAP Catalog (AMD)

Starting with Oracle Database 12c, the Common Warehouse Metamodel (CWM) standard is desupported for the OLAP catalog (AMD).

CWM standard support was deprecated in Oracle Database 11g Release 2 (11.2). If your existing database has CWM metadata in the OLAP catalog, and you upgrade to Oracle Database 12c, then the upgraded database has the AMD component. If the database you upgrade does not have the AMD component, then the upgraded Oracle Database 12c database also does not have the AMD component, because new installations for Oracle Database 12c do not include AMD. If your database has the AMD component, and you want to remove it, then run the `catnoamd.sql` script that is located in the path*`ORACLE_HOME`*`/olap/admin/catnoamd.sql`. You can run this script either before or after you complete your upgrade.

> **Note:**
>
> If the OLAP catalog exists in the database you are upgrading, then you can see `AMD OLAP Catalog OPTION OFF` and invalid CWM OLAP objects. You can safely ignore the invalid OLAP objects, because they are not needed.

See Also:

- *Oracle OLAP Java API Developer's Guide* for information about OLAP Java API metadata
- *Oracle OLAP User's Guide* for more information about the OLAP option of Oracle Database and online analytic processing

Desupport of CSSCAN and CSALTER for Oracle Globalization

Oracle Database 12*c* includes Oracle Database Migration Assistant for Unicode (DMU), and Oracle is desupporting the legacy database tools CSSCAN and CSALTER.

DMU provides a complete end-to-end Unicode migration solution for database administrators. Starting with Oracle Database 12*c*, DMU is included with Oracle Database. The CSSCAN and CSALTER tools are no longer included or supported.

Desupport of Oracle Net Connection Pooling

In Oracle Database 12*c*, Oracle Net connection pooling is no longer supported.

Oracle Net connection pooling was deprecated in Oracle Database 11*g*. This deprecation included the `DISPATCHERS` attributes `TICKS`, `SESSIONS`, and `CONNECTIONS`.

See Also:

Oracle Database Net Services Administrator's Guide for information about configuring dispatchers

Oracle Database Net Services Administrator's Guide for information about Oracle Connection Manager Parameters (cman.ora)

Desupport of Oracle Names

Oracle Names was desupported as a naming method in Oracle Database 11*g*. You must migrate to directory naming.

See Also:

Oracle Database Net Services Administrator's Guide for additional information about migrating to directory naming

Desupport of Oracle Net Listener Password

In Oracle Database 12c, the Oracle Net Listener password feature is no longer supported.

This change does not cause a loss of security, because Oracle Database enforces authentication through local operating system authentication.

Related Topics

- *Oracle Database Net Services Reference*

Desupport of SQLNET.KERBEROS5_CONF_MIT Parameter for Oracle Net Services

The `SQLNET.KERBEROS5_CONF_MIT` networking parameter is no longer needed and is not supported in `sqlnet.ora`.

By default, the value of `SQLNET.KERBEROS5_CONF_MIT` is set to `FALSE`. This parameter setting has no affect on the Kerberos configuration. In previous releases, when `SQLNET.KERBEROS5_CONF_MIT` is set to `TRUE`, the parameter set parsing in a format as specified by `MIT Kerberos 5`. However, this parameter setting is no longer required. Starting with Oracle Database 12*c*, only `MIT Kerberos 5` configuration is supported.

> **See Also:**
>
> *Oracle Database Net Services Reference* for information about Kerberos parameters for the `sqlnet.ora` file

Desupport of Oracle Names Control Utility for Oracle Net Services

The Oracle Names Control Utility is desupported. Starting with Oracle Database 10*g*, it is not available.

Desupport of the Oracle Names Control Utility includes desupporting all the related control utility commands. Oracle Database clients cannot use a Names Server to resolve connect strings. Migrate your applications to Oracle Internet Directory with LDAP directory naming.

> **See Also:**
>
> *Oracle Database Net Services Reference* for information about configuring the directory naming method

Desupport of CTXXPATH in Oracle Text and Oracle XML DB

The `CTXSYS.CTXXPATH` index is desupported, starting with Oracle Database 12c Release 1 (12.1).

The desupport of `CTXSYS.CTXXPATH` does not affect `CTXCAT`. Use `XMLIndex` indexes instead.

Desupport of ALTER INDEX OPTIMIZE for Text Indexes

The ALTER INDEX OPTIMIZE `[token index_token | fast | full [maxtime (time | unlimited)]` operation is not supported for Oracle Database 12*c*.

To optimize your index, use CTX_DDL.OPTIMIZE_INDEX.

> **See Also:**
>
> *Oracle Text Reference* for information about OPTIMIZE_INDEX

Desupport of SYNC [MEMORY memsize] for Text Indexes

The `SYNC [MEMORY memsize]` operation is not supported for Oracle Database 12c

To synchronize your index, use `CTX_DDL.SYNC_INDEX`.

Related Topics

- *Oracle Text Reference*

A

Oracle Database Upgrade Utilities

Oracle Upgrade utility scripts help to carry out Oracle Database upgrades.

- Scripts for Upgrading Oracle Database
  Oracle provides a set of tools and scripts for upgrades that you run before, during, and after upgrading.
- Using the Pre-Upgrade Information Tool for Oracle Database
  Review these topics to understand and to use the Pre-Upgrade information tool (`preupgrade.jar`).

Scripts for Upgrading Oracle Database

Oracle provides a set of tools and scripts for upgrades that you run before, during, and after upgrading.

> **Note:**
>
> Some of the scripts Oracle provides cannot be run in UPGRADE mode.

The following table lists the various scripts and tools with a description for each.

Table A-1 Upgrade, Post-Upgrade, and Downgrade Scripts

Script	Description
`catcon.pl`	Must be run in UPGRADE mode. This script is used when upgrading a CDB.
`catctl.pl`	Parallel Upgrade Utility (`catctl.pl`, and the shell script `dbupgrade`). You must run these scripts in UPGRADE mode. In Oracle Database 12*c* and later releases, the Parallel Upgrade Utility replaces the SQL Upgrade Utility `catupgrd.sql`. With Parallel Upgrade Utility, you can run upgrade scripts and processes in parallel. This capability takes full advantage of your server CPU capacity, and can shorten upgrade time. DBUA uses this tool.
`dbdowngrade`	Shell script utility that calls the `catdwgrd.sql` script, and ensures that calls to the `catcon.pl` script use the recommended values for the downgrade. This feature helps to reduce potential errors due to excessive threads being spawned during the upgrade. It is particularly helpful with multitenant architecture (CDB) downgrades. Using this script is Oracle's recommended downgrade procedure.

Table A-1 (Cont.) Upgrade, Post-Upgrade, and Downgrade Scripts

Script	Description
`dbupgrade`, `dbupgrade.cmd`	Shell scripts that call the `catctl.pl` script. These shell scripts are available with Oracle Database 12*c* release 2 (12.2) and later releases. The scripts enable you to enter the `dbupgrade` command at the shell command prompt. You can either run these scripts with default values, or you can run them with the same input parameters that you use to run `catctl.pl` from the Perl prompt. Use `dbupgrade` for Linux and UNIX systems, and `dbupgrade.cmd` for Windows systems.
`catdwgrd.sql`	This is the downgrade script, which is used in the procedure to downgrade to the earlier release from which you upgraded.
`catnoamd.sql`	Do not run in UPGRADE mode. Run this script only after upgrading. The `catnoamd.sql` script drops the OLAP catalog component, also referred to as AMD.
`catnoexf.sql`	The upgrade process runs this script for you; however, you can run the `catnoexf.sql` script before the upgrade. Do not run this script in UPGRADE mode. The `catnoexf.sql` script removes the Expression Filter (EXF) and Database Rules Manager (RUL) components, which are desupported in this release.
`catuppst.sql`	You must run this script, either through DBUA or manually, if you perform a manual upgrade. DBUA automatically runs `catuppst.sql`. You only must run this script separately for manual upgrades. Do not run this in UPGRADE mode. Run `catuppst.sql`, located in the *`ORACLE_HOME`*`/rdbms/admin` directory, to perform remaining upgrade actions that do not require the database to be in `UPGRADE` mode. If an Oracle bundle patch or patch set update (PSU or BP) is installed in the Oracle home, then this script automatically applies that patch set update to the database. **Caution:** If you perform a manual upgrade, and you do not run `catuppst.sql`,, then your database suffers performance degradation over time.
`catuptabdata.sql`	The `catuptabdata.sql` script is run automatically by `catuppst.sql` to run ALTER TABLE UPGRADE on any Oracle-Maintained tables that are affected by changes to Oracle-Maintained types during the upgrade. You can run the `catuptabdata.sql` script manually to upgrade any Oracle-Maintained tables that require upgrading because of changes to any Oracle-Maintained types. You must run the command with a user account that is granted the SYSDBA system privileges, and that is connected AS SYSDBA.

Table A-1 (Cont.) Upgrade, Post-Upgrade, and Downgrade Scripts

Script	Description
`emdwgrd`	Do not run this script in UPGRADE mode. The `emdwgrd` utility saves Oracle Enterprise Manager Database Control (DB Control) configuration files and data for your release 11*g* database. If it is possible that you want to downgrade Oracle Database 18c to Oracle Database 11*g* (11.2), and you want to restore Oracle DB Control, then you must run `emdwgrd` before you upgrade. **See Also:** Oracle Support document ID 1484775.1.
`emremove.sql`	The `emremove.sql` script drops the Oracle Enterprise Manager-related schemas and objects. Use this script to manually remove DB Control. Running `emremove.sql` before starting the upgrade process minimizes downtime. This is an optional pre-upgrade step because the Parallel Upgrade Utility and DBUA automatically run this script. **Caution**: If you want to preserve the DB Control configuration and data to have the option of downgrading and restoring DB Control, then you must first follow the procedure for using `emdwgrd` .
`olspreupgrade.sql`	Do not run this script in UPGRADE mode. The `olspreupgrade.sql` script is a preprocessing script required for upgrading a database earlier than Oracle Database release 12.1 that uses Oracle Label Security (OLS) and Oracle Database Vault.
`postupgrade_fixups.sql`	The `postupgrade_fixups.sql` script is supplied with Oracle Database. Run `postupgrade_fixups.sql` after upgrading. DBUA runs this script automatically; however, you can run it any time after upgrading.
`preupgrd.sql`	(Obsolete in 12.2 and later releases) The `preupgrd.sql` Pre-Upgrade Information Tool is supplied with Oracle Database 12c release 1 (12.1). Run `preupgrd.sql` any time before upgrading to analyze your database. The Pre-Upgrade Information Tool provides a preview of the items that DBUA checks and generates scripts, called Fixup scripts, that you can run to resolve issues that are flagged in the source database. In release 12.1, `preupgrd.sql` script replaces the `utlu121i.sql` script.) In release 12.2, `preupgrade.jar` replaces `preupgrd.sql`.
`preupgrade.jar`	The `preupgrade.jar` Pre-Upgrade Information Tool is supplied with Oracle Database 12*c* release 2 (12.2). The `preupgrade.jar` has the same capabilities as `preupgrd.sql`, which `preupgrade.jar` replaces.
`utlrp.sql`	Use `utlrp.sql` to recompile stored PL/SQL and Java code. DBUA runs this script automatically. When you upgrade manually, you must run this script.

Table A-1 (Cont.) Upgrade, Post-Upgrade, and Downgrade Scripts

Script	Description
`utlusts.sql`	The `utlusts.sql` Post-Upgrade Status Tool is supplied with Oracle Database and displays the version and elapsed upgrade time for each component in `DBA_REGISTRY`. The Post-Upgrade Status Tool can be run any time after upgrading the database.
`utluptabdata.sql`	Run the `utluptabdata.sql` script after upgrades to perform an ALTER TABLE UPGRADE command on any user tables that depend on Oracle-Maintained types that changed during the upgrade. You must run the `utluptabdata.sql` script either with a user account that is assigned the ALTER TABLE system privilege for all of the tables that you want to upgrade, or by a user account with SYSDBA system privileges that is connected AS SYSDBA). User tables are not automatically upgraded to new versions of types during the database upgrade, so that you can run the upgrade with user tablespaces set to READ ONLY.

Using the Pre-Upgrade Information Tool for Oracle Database

Review these topics to understand and to use the Pre-Upgrade information tool (`preupgrade.jar`).

> **Note:**
>
> The Pre-Upgrade information tool is a legacy utility. Oracle recommends that you use the AutoUpgrade utility instead.

- About the Pre-Upgrade Information Tool
  To determine if your system is ready for upgrading, you can use the legacy Pre-Upgrade Information Tool (`preupgrade.jar`)
- Preupgrade Scripts Generated By the Pre-Upgrade Information Tool
  You can run preupgrade scripts that the Pre-Upgrade Information Tool generates to fix many issues before you upgrade to the new Oracle Database release.
- Postupgrade Scripts Generated By the Pre-Upgrade Information Tool
  After the upgrade, you can run the postupgrade scripts that the Pre-Upgrade Information Tool generates to complete fixups of your upgrade target database.
- Setting Up Environment Variables for the Pre-Upgrade Information Tool
  Before you run the Pre-Upgrade Information Tool, set up the user environment variables for the Oracle user that runs the tool.

- Pre-Upgrade Information Tool (preupgrade.jar) Command
  Use Pre-Upgrade Information Tool (`preupgrade.jar`) commands to check your system before upgrades.
- Output of the Pre-Upgrade Information Tool
  The Pre-Upgrade Information Tool (`preupgrade.jar`) creates fixup scripts and log files in the output directory that you specify with the DIR command-line option.
- Pre-Upgrade Information Tool Output Example
  In this example, you can see how the Pre-Upgrade Information Tool displays recommended fixes, but does not carry out fixes automatically.
- Pre-Upgrade Information Tool Warnings and Recommendations for Oracle Database
  Analyze any Pre-Upgrade Information Tool warnings before you upgrade to the new release of Oracle Database. For each item that the tool reports, it provides you with information about how to fix the issue or warning.

About the Pre-Upgrade Information Tool

To determine if your system is ready for upgrading, you can use the legacy Pre-Upgrade Information Tool (`preupgrade.jar`)

To help to ensure a successful upgrade, Oracle strongly recommends that you run the AutoUpgrade Utility in Analyze processing mode, and to use the AutoUpgrade Utility Fixup, and then complete your upgrade with the method that you prefer. However, you can continue to perform these tasks by using the Pre-Upgrade Information Tool before you begin your upgrade.

If you use the legacy Pre-Upgrade Information tool, then after the upgrade is complete, you can use the postupgrade scripts that it generates to help to assist you with fixing any issues that the tool discovers. To obtain the latest updates, Oracle recommends that you download the most recent version of the tool from My Oracle Support Note 884522.1.

You can run the tool from the operating system command line. The Pre-Upgrade Information Tool creates preupgrade scripts, which fix some issues before you start an upgrade, and postupgrade scripts, which fix some issues after an upgrade is completed.

The Pre-Upgrade Information Tool (`preupgrade.jar`) creates the following files:

- The log file `preupgrade.log`.

  The log file contains the output of the Pre-Upgrade Information Tool.
- The `preupgrade_fixups_`*`pdbname`*`.sql` for PDBs, where *`pdbname`* is the name of the PDB).

  Before you run the upgrade, you can run the preupgrade fixups script manually in SQL*Plus to resolve many of the issues identified by the preupgrade tool.
- The `postupgrade_fixups_`*`pdbname`*`.sql` (for PDBs, where *`pdbname`* is the name of the PDB) or `postupgrade_fixups.sql` script (Non-CDB databases).

  You can run this script to fix issues after the database upgrade is completed.

Related Topics

- My Oracle Support Note 884522.1

Preupgrade Scripts Generated By the Pre-Upgrade Information Tool

You can run preupgrade scripts that the Pre-Upgrade Information Tool generates to fix many issues before you upgrade to the new Oracle Database release.

The location of the `preupgrade_fixups.sql` and log files depends on how you set output folders, or define the Oracle base environment variable.

If you specify an output directory by using the `dir` option with the Pre-Upgrade Information Tool, then the output logs and files are placed under that directory in the file path /`cfgtoollogs`/*`dbunique_name`*/`preupgrade`, where *`dbunique_name`* is the name of your source Oracle Database. If you do not specify an output directory when you run the Pre-Upgrade Information Tool, then the output is directed to one of the following default locations:

- If you do not specify an output directory with `DIR`, but you have set an Oracle base environment variable, then the generated scripts and log files are created in the following file path:

  `Oracle-base``/cfgtoollogs/`*`dbunique_name`*`/preupgrade`

- If you do not specify an output directory, and you have not defined an Oracle base environment variable, then the generated scripts and log files are created in the following file path:

  `Oracle-home``/cfgtoollogs/`*`dbunique_name`*`/preupgrade`

The fixup scripts that the Pre-Upgrade Information Tool creates depend on whether your source database is a Non-CDB database, or a CDB database:

- A log file (`preupgrade.log`).

  The log file contains log output for the Pre-Upgrade Information Tool.

- Pre-upgrade fixups SQL scripts, depending on your source database type:
  - **Non-CDB**: `preupgrade_fixups.sql`
  - **CDB**: Two different sets of scripts:
    1. `preupgrade_fixups.sql`: A consolidated script for all PDBs.
    2. Multiple `preupgrade_fixups_`*`pdbname`*`.sql` scripts, where*`pdbname`* is the name of the PDB for which a script is generated: Individual scripts, which you run on specific PDBs.

  Run the scripts either by using `catcon.pl`, or by using SQL*Plus commands. You must run these scripts to fix issues before you start the database upgrade. The scripts resolve many of the issues identified by the preupgrade tool.

  Each issue that the scripts identify includes a description of the problem, and a task that you can carry out to resolve the problem. The preupgrade tool itself does not make changes to your database to correct errors. However, you can run the scripts that it generates to correct identified errors. The scripts fix only those issues that an automated script can fix safely. Preupgrade issues that the automated script cannot fix safely typically require DBA knowledge of user applications. You can address those issues manually.

Postupgrade Scripts Generated By the Pre-Upgrade Information Tool

After the upgrade, you can run the postupgrade scripts that the Pre-Upgrade Information Tool generates to complete fixups of your upgrade target database.

The Pre-Upgrade Information Tool generates postupgrade fixup scripts, which you can run after the upgrade to fix issues that can be fixed after the upgrade.

The location of the postupgrade SQL scripts and log files depends on how you set output folders, or define the Oracle base environment variable. The postupgrade fixup scripts are placed in the same directory path as the preupgrade fixup scripts.

If you specify an output directory by using the `dir` option with the Pre-Upgrade Information Tool, then the output logs and files are placed under that directory in the file path /`cfgtoollogs`/*`dbunique_name`*/`preupgrade`, where *`dbunique_name`* is the name of your source Oracle Database. If you do not specify an output directory when you run the Pre-Upgrade Information Tool, then the output is directed to one of the following default locations:

- If you do not specify an output directory with `DIR`, but you have set an Oracle base environment variable, then the generated scripts and log files are created in the following file path:

  `Oracle-base``/cfgtoollogs/`*`dbunique_name`*`/preupgrade`

- If you do not specify an output directory, and you have not defined an Oracle base environment variable, then the generated scripts and log files are created in the following file path:

  `Oracle-home``/cfgtoollogs/`*`dbunique_name`*`/preupgrade`

The postupgrade fixup scripts that the Pre-Upgrade Information Tool creates depend on whether your source database is a Non-CDB database, or a CDB database:

- **Non-CDB**: `postupgrade_fixups.sql`
- **CDB**: Two different sets of scripts:
  1. `postupgrade_fixups.sql`: A consolidated script for all PDBs
  2. Multiple `postupgrade_fixups_`*`pdbname`*`.sql` scripts, where*`pdbname`* is the name of the PDB for which a script is generated: Individual scripts, which you run on specific PDBs.

Postupgrade issues that the automatic script cannot fix safely typically require DBA knowledge of user applications. You can address those issues manually.

Guidelines for Running Postupgrade Fixup Scripts for Non-CDB Databases

Oracle recommends that when you run the postupgrade scripts, you set the system to spool results to a log file so you can read the output. However, do not spool results to the `admin` directory:

After you run postupgrade scripts, you can run the Post-Upgrade Status Tool to check the status of your server.

Related Topics

- Running Postupgrade Fixup Scripts

Setting Up Environment Variables for the Pre-Upgrade Information Tool

Before you run the Pre-Upgrade Information Tool, set up the user environment variables for the Oracle user that runs the tool.

You must set up the user environment variables for the Pre-Upgrade Information Tool. This example shows how to use shell commands to set up user environment variables to point to an earlier release Oracle home. For multitenant architecture upgrades, you must also open up all the PDBs that you want the tool to analyze.

In this example, the operating system is Linux or Unix, the system identifier is `sales01`, and the earlier release Oracle home path is `/u01/app/oracle/product/12.1.0/dbhome_1`

1. Log in as the Oracle installation owner (`oracle`).
2. Set up the user environment variables to point to the earlier release Oracle home that you want to upgrade.

   For example:

   ```
   $ export ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1
   $ export ORACLE_BASE=/u01/app/oracle
   $ export ORACLE_SID=sales01
   $ export PATH=.:$ORACLE_HOME/bin:$PATH
   ```

3. (Multitenant architecture upgrades): If you are running the Pre-Upgrade Information Tool in a CDB, then use the command `alter pluggable database all open` to open the PDBs so that the tool can analyze them.

   For example: Log in to the database and enter the following command to open all PDBs:

   ```
   $ sqlplus \ as SYSDBA
   .
   .
   .
   SQL> alter pluggable database all open;
   ```

Pre-Upgrade Information Tool (preupgrade.jar) Command

Use Pre-Upgrade Information Tool (`preupgrade.jar`) commands to check your system before upgrades.

Prerequisites

The Pre-Upgrade Information Tool is in the new release Oracle home, in the file path `ORACLE_HOME/rdbms/admin/preupgrade.jar`. Oracle has configured it with the system checks necessary for the new Oracle Database release. However, the checks that the tool performs are carried out on the earlier release Oracle Database home. Set up the Oracle user environment variables so that they point to the earlier release Oracle home.

Run the Pre-Upgrade Information Tool by using the Java version in your earlier release Oracle home. For multitenant architecture (CDB and PDB) upgrades, open up all the PDBs that you want the tool to analyze before you run the tool.

Set the environment variables for your user account to point to the earlier release ORACLE_HOME, ORACLE_BASE, and ORACLE_SID.

File Path

The `preupgrade.jar` file is located in the new Oracle home:

New_release_Oracle_home`/rdbms/admin/preupgrade.jar`

You can also copy the `preupgrade.jar` binaries to a path of your choosing. For example:

```
/tmp/preupgrade.jar
```

Syntax

$*Earlier_release_Oracle_home*`/jdk/bin/java -jar $`*New_release_Oracle_home*`/rdbms/admin/preupgrade.jar [FILE|TERMINAL] [TEXT|XML] [DIR` *output_dir*`]`

Options

Command Option	Description
`FILE|TERMINAL`	Script output location. Use `FILE` to direct script output to a file. Use `TERMINAL` to direct output to the terminal. If you do not specify a value, then the default is `FILE`. If you specify `TERMINAL`, then screen output is directed to the display, and scripts and logs are placed in the output directory path.
`TEXT|XML`	Output type. Use `XML` to specify XML output. If you do not specify an output type, then the default is `TEXT`.
`DIR` *output_dir*	Directs the output to a specific directory. If you do not specify an output directory with the `DIR` option, then the output is directed to one of the following default locations: • If you do not specify an output directory with `DIR`, but you define an *ORACLE_BASE* environment variable, then the generated scripts and log files are created. • If you do not specify an output directory, and *ORACLE_BASE* is not defined, then the generated scripts and log files are created in the following path: *ORACLE_HOME*`/cfgtoollogs/`*dbunique_name*`/preupgrade`
`-c "pdb1 pdb2 pdb3"` (Windows) `-c 'pdb1 pdb2 pdb3'` (Linux and UNIX)	Specifies a list of containers inside a CDB that you want to include for processing (an allow list). Provide a space-delimited list of PDBs that you want processed. To specify the list, use single quotes on Linux and UNIX operating systems, and use double quotes on Windows systems. If you do not specify either `-c` or `-C`, then all PDBs in a CDB are processed.

Command Option	Description
`-C "pdb1 pdb2 pdb3"` (Windows) `-C 'pdb1 pdb2 pdb3'` (Linux and UNIX)	Specifies a list of containers inside a CDB that you want to exclude from processing (a block list). Provide a space-delimited list of PDBs that you want to exclude from processing. To specify the list, use single quotes on Linux and UNIX operating systems, and use double quotes on Windows systems. If you do not specify either `-c` or `-C`, then all PDBs in a CDB are processed.
`-loadonly`	Loads the `DBMS_PREUP` package into the database when it is in `READ WRITE` mode, without carrying out any other action. You can use this parameter to prepare a given Non-CDB or CDB database so that the `DBMS_PREUP` package is loaded when you run the Pre-Upgrade Information Tool, and the DB (DB or Container) is in `READ-ONLY` mode. If you want use the tool to analyze a database in read-only mode, then you must use this command to load the `DBMS_PREUP` package into the database while it is in `READ WRITE` mode, before you set it to `READ-ONLY` mode. You can also use this parameter with a Read-Only Standby database, in which you load the package in the primary database, and run the package in the standby.
`-p` *`password`*	Provides the password for the user. If you do not use operating system authentication to connect to the database, then use the `-p` option to specify a password on the command line. If a username is specified on the command line with `-u`, but no password specified with `-p`, then the tool prompts you for a password.
`-u` *`username`*	Provides the user name of the user that you want to use to connect as SYSDBA to the database that you want to check. Use this option only if you do not use operating system authentication to connect to the database For example, You log in as a user that is not a member of the OSDBA group for the database that you want to check. In that case, the user account does not have operating system authentication privileges for the SYSDBA system privilege. Use the `-u` and `-p` option to provide data dictionary authentication to log in as a user with SYSDBA system privileges.
`-oh` *`oracle_home`*	Specifies an Oracle home that you want to check. Provide the path of the Oracle home that you want to check. If you do not specify an Oracle home path to check, then the Pre-Upgrade Information Tool defaults to the path specified by the user environment variable for the Oracle home. That variable is `$ORACLE_HOME` on Linux and Unix systems, and `%ORACLE_HOME%` on Windows systems.
`-sid` *`system_identifier`*	Specifies an Oracle system identifier that you want to check. Provide the ORACLE_SID of the database that you want to check.
`-help`	Displays the command-line syntax help text.

Example A-1 Non-CDB In the Source Oracle Home Example

1. Set your user environment variables to point to the earlier release Oracle home.

```
$ export ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1
$ export ORACLE_BASE=/u01/app/oracle
$ export ORACLE_SID=sales01
$ export PATH=.:$ORACLE_HOME/bin:$PATH
```

2. Run the new release Oracle Database Pre-Upgrade Information Tool on the earlier release Oracle Database server using the environment settings you have set to the earlier release Oracle home. For example:

```
$ORACLE_HOME/jdk/bin/java -jar /u01/app/oracle/product/19.0.0/rdbms/
admin/preupgrade.jar TERMINAL TEXT
```

Example A-2 CDB in a Source Oracle Home

1. Open all the pluggable databases

```
SQL> alter pluggable database all open;
```

2. Set your user environment variables to point to the earlier release Oracle home.

```
$ export ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1
$ export ORACLE_BASE=/u01/app/oracle
$ export ORACLE_SID=sales01
$ export PATH=.:$ORACLE_HOME/bin:$PATH
```

3. Run the Pre-Upgrade Information Tool with an inclusion list, using the `-c` option. In this example, the inclusion list is PDB1 and PDB2, and the command is run on a Linux or UNIX system. The output of the command is displayed to the terminal, and the output is displayed as text.

```
$ORACLE_HOME/jdk/bin/java -jar /u01/app/oracle/product/19.0.0/
dbhome_1/rdbms/admin/preupgrade.jar TERMINAL TEXT -c 'pdb1 pdb2'
```

Output of the Pre-Upgrade Information Tool

The Pre-Upgrade Information Tool (`preupgrade.jar`) creates fixup scripts and log files in the output directory that you specify with the DIR command-line option.

When you run the Pre-Upgrade Information Tool, it generates the following files inside the directory that you specify as the output directory.

Log File (preupgrade.log)

The file `preupgrade.log` is the report that the Pre-Upgrade Information Tool generates whenever you run the command with the FILE option. The log file contains all the tool recommendations and requirements for upgrade. The log file is located in the following path, where *`timestamp`* is the date and time when the command is run: `$ORACLE_BASE/cfgtoollogs/dbua/upgrade`*`timestamp`*`/SID/`. If you run the command with the TERMINAL option, then the content of this file is output to the

display. Refer to the section "Pre-Upgrade Information Tool Output Example" for an example of a log file.

XML DBUA File (upgrade.xml)

If you specify XML file output on the Pre-Upgrade Information Tool command line, then it generates the `upgrade.xml` file instead of `preupgrade.log`.

Preupgrade Fixup File (preupgrade_fixups.sql) and Postupgrade Fixup File (postupgrade_fixups.sql)

The Pre-Upgrade Information Tool identifies issues that can block or hinder an upgrade.

Some issues require a DBA to resolve, because it is not possible for the automated script to understand the specific goals of your application. However, other issues do not present any difficulty in resolving. In these cases, the Pre-Upgrade Information Tool automatically generates scripts that contain the SQL statements necessary to resolve the issues. Using these scripts can perform, track, and simplify the work that DBAs must do to resolve potential upgrade issues. The SQL statements that resolve issues before upgrade are placed in the `preupgrade_fixups.sql` script. The SQL statements that resolve issues after upgrade are placed in the `postupgrade_fixups.sql` script. When you run the Pre-Upgrade Information tool on a multitenant architecture Oracle Database, you can run the consolidated scripts `preupgrade_fixups.sql` script and `postupgrade_fixups.sql` across all the containers. Run the consolidated scripts using `catcon.pl`.

Both of these fixup files are generated in the output directory that you specify with the Pre-Upgrade Information Tool `DIR` command-line option.

The script carries out the following steps to resolve pre-upgrade or post-upgrade issues:

1. For each issue that the Pre-Upgrade Information Tool identifies, it reruns the individual Pre-Upgrade Information Tool check again, to determine if the issue is still present.
2. If the issue is present, then the tool does one of the following, depending on whether there is a fixup routine for the issue:
   - If there is an Oracle-supplied fixup routine for that issue, then the script executes that fixup routine. It then reruns the Pre-Upgrade Information Tool check again, to confirm that the issue is resolved. If the issue is resolved, then the script displays a message that the issue is resolved.
   - If there is no Oracle-supplied fixup routine, then the script displays a message that the issue is still present.

Review the issues that the script indicates are still present after it completes its checks. Resolve all the issues marked `Required` before upgrading. You can choose to resolve recommended fixes and informational fixes. You can rerun the `preupgrade_fixups.sql` and `postupgrade_fixups.sql` scripts as many times as you want to. You can use the scripts as a progress report to track remaining issues as part of your issue resolution plan.

After you resolve issues identified by the Preupgrade fixup and Postupgrade fixup scripts, Oracle recommends that you rerun the Pre-Upgrade Information Tool (`preupgrade.jar`) one more time before upgrade. Running the tool one more time helps to confirm that you have resolved all issues that you intend to address.

If you are checking a multitenant environment system (CDBs and PDBs), then the `preupgrade_fixups.sql` and `postupgrade_fixups.sql` scripts contain a rollup of the fixup code for all the individual PDBs. They are coded so that only the code for the current PDB runs. You can run the same script in each PDB. The script fixes only the issues in that PDB. As a result, it is easy to use the `preupgrade_fixups.sql` and `postupgrade_fixups.sql` with `catcon.pl` to run fixups across an entire CDB.

Related Topics

- Pre-Upgrade Information Tool Output Example
- Scripts for Upgrading Oracle Database

Pre-Upgrade Information Tool Output Example

In this example, you can see how the Pre-Upgrade Information Tool displays recommended fixes, but does not carry out fixes automatically.

You have control over how and when the fixup scripts are run.

The following example shows the output that is generated and written to `preupgrade.log` by running the Oracle Database 18c Pre-Upgrade Information Tool on a release 12.2.0.1 CDB:

```
$ java -jar preupgrade.jar TEXT TERMINAL
Report generated by Oracle Database Pre-Upgrade Information Tool Version
18.0.0.0.0 on 2018-06-13T15:08:45

Upgrade-To version: 18.0.0.0.0

======================================
Status of the database prior to upgrade
======================================
      Database Name:  CDB2
     Container Name:  CDB$ROOT
       Container ID:  1
            Version:  12.2.0.1.0
         Compatible:  12.2.0
          Blocksize:  8192
           Platform:  Linux x86 64-bit
      Timezone File:  26
  Database log mode:  NOARCHIVELOG
           Readonly:  FALSE
            Edition:  EE

  Oracle Component                       Upgrade Action    Current
Status
  ----------------                       --------------
--------------
  Oracle Server                          [to be upgraded]
VALID
  JServer JAVA Virtual Machine           [to be upgraded]
VALID
  Oracle XDK for Java                    [to be upgraded]
VALID
  Real Application Clusters              [to be upgraded]  OPTION
```

```
OFF
  Oracle Workspace Manager                [to be upgraded]
VALID
  Oracle Label Security                   [to be upgraded]
VALID
  Oracle XML Database                     [to be upgraded]
VALID
  Oracle Java Packages                    [to be upgraded]
VALID

==============
BEFORE UPGRADE
==============

  REQUIRED ACTIONS
  ================
  None

  RECOMMENDED ACTIONS
  ===================
  1.  Run 12.2.0.1.0 $ORACLE_HOME/rdbms/admin/utlrp.sql to recompile
invalid
      objects.  You can view the individual invalid objects with

        SET SERVEROUTPUT ON;
        EXECUTE DBMS_PREUP.INVALID_OBJECTS;

      3 objects are INVALID.

      There should be no INVALID objects in SYS/SYSTEM or user schemas
before
      database upgrade.

  2.  Review and remove any unnecessary HIDDEN/UNDERSCORE parameters.

      The database contains the following initialization parameters
whose name
      begins with an underscore:

      _exclude_seed_cdb_view

      Remove hidden parameters before database upgrade unless your
application
      vendors and/or Oracle Support state differently.  Changes will
need to be
      made in the pfile/spfile.

  3.  (AUTOFIXUP) Gather stale data dictionary statistics prior to
database
      upgrade in off-peak time using:

        EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;

      Dictionary statistics do not exist or are stale (not up-to-date).
```

```
      Dictionary statistics help the Oracle optimizer find efficient SQL
      execution plans and are essential for proper upgrade timing.
Oracle
      recommends gathering dictionary statistics in the last 24 hours
before
      database upgrade.

      For information on managing optimizer statistics, refer to the
12.2.0.1
      Oracle Database SQL Tuning Guide.

  INFORMATION ONLY
  ================
  4.  To help you keep track of your tablespace allocations, the
following
      AUTOEXTEND tablespaces are expected to successfully EXTEND during
the
      upgrade process.

                                                  Min Size
      Tablespace                     Size         For Upgrade
      ----------                  ----------      -----------
      SYSAUX                         550 MB          616 MB
      SYSTEM                         700 MB         1117 MB
      TEMP                            22 MB          150 MB
      UNDOTBS1                       315 MB          433 MB

      Minimum tablespace sizes for upgrade are estimates.

  5.  No action needed.

      Using default parallel upgrade options, this CDB with 1 PDBs will
first
      upgrade the CDB$ROOT, and then upgrade at most 1 PDBs at a time
using 2
      parallel processes per PDB.

      The number of PDBs upgraded in parallel and the number of parallel
      processes per PDB can be adjusted as described in Database
Upgrade Guide.

  ORACLE GENERATED FIXUP SCRIPT
  =============================
  All of the issues in database CDB2 container CDB$ROOT
  which are identified above as BEFORE UPGRADE "(AUTOFIXUP)" can be
resolved by
  executing the following from within the container

    SQL>@/u01/app/oracle/cfgtoollogs/CDB2/preupgrade/
preupgrade_fixups.sql

==============
AFTER UPGRADE
==============
```

```
  REQUIRED ACTIONS
  ================
  None

  RECOMMENDED ACTIONS
  ===================
  6.  Upgrade the database time zone file using the DBMS_DST package.

      The database is using time zone file version 26 and the target
18.0.0.0.0
      release ships with time zone file version 31.

      Oracle recommends upgrading to the desired (latest) version of
the time
      zone file.  For more information, refer to "Upgrading the Time
Zone File
      and Timestamp with Time Zone Data" in the 18.0.0.0.0 Oracle
Database
      Globalization Support Guide.

  7.  (AUTOFIXUP) Gather dictionary statistics after the upgrade using
the
      command:

        EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;

      Oracle recommends gathering dictionary statistics after upgrade.

      Dictionary statistics provide essential information to the Oracle
      optimizer to help it find efficient SQL execution plans. After a
database
      upgrade, statistics need to be re-gathered as there can now be
tables
      that have significantly changed during the upgrade or new tables
that do
      not have statistics gathered yet.

  8.  Gather statistics on fixed objects after the upgrade and when
there is a
      representative workload on the system using the command:

        EXECUTE DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;

      This recommendation is given for all preupgrade runs.

      Fixed object statistics provide essential information to the
Oracle
      optimizer to help it find efficient SQL execution plans.  Those
      statistics are specific to the Oracle Database release that
generates
      them, and can be stale upon database upgrade.

      For information on managing optimizer statistics, refer to the
12.2.0.1
      Oracle Database SQL Tuning Guide.
```

```
  ORACLE GENERATED FIXUP SCRIPT
  ============================
  All of the issues in database CDB2 container CDB$ROOT
  which are identified above as AFTER UPGRADE "(AUTOFIXUP)" can be
resolved by
  executing the following from within the container

    SQL>@/u01/app/oracle/cfgtoollogs/CDB2/preupgrade/
postupgrade_fixups.sql


Report generated by Oracle Database Pre-Upgrade Information Tool Version
18.0.0.0.0 on 2018-06-13T15:08:58

Upgrade-To version: 18.0.0.0.0

======================================
Status of the database prior to upgrade
======================================
      Database Name:  CDB2
     Container Name:  PDB$SEED
       Container ID:  2
            Version:  12.2.0.1.0
         Compatible:  12.2.0
          Blocksize:  8192
           Platform:  Linux x86 64-bit
      Timezone File:  26
  Database log mode:  NOARCHIVELOG
           Readonly:  TRUE
            Edition:  EE

  Oracle Component                       Upgrade Action    Current
Status
  ----------------                       --------------
--------------
  Oracle Server                          [to be upgraded]
VALID
  Real Application Clusters              [to be upgraded]  OPTION
OFF
  Oracle Workspace Manager               [to be upgraded]
VALID
  Oracle XML Database                    [to be upgraded]
VALID

==============
BEFORE UPGRADE
==============

  REQUIRED ACTIONS
  ================
  None

  RECOMMENDED ACTIONS
  ===================
```

```
  1.  Run 12.2.0.1.0 $ORACLE_HOME/rdbms/admin/utlrp.sql to recompile
invalid
      objects.  You can view the individual invalid objects with

        SET SERVEROUTPUT ON;
        EXECUTE DBMS_PREUP.INVALID_OBJECTS;

      6 objects are INVALID.

      There should be no INVALID objects in SYS/SYSTEM or user schemas
before
      database upgrade.

  2.  Review and remove any unnecessary HIDDEN/UNDERSCORE parameters.

      The database contains the following initialization parameters
whose name
      begins with an underscore:

      _exclude_seed_cdb_view

      Remove hidden parameters before database upgrade unless your
application
      vendors and/or Oracle Support state differently.  Changes will
need to be
      made in the pfile/spfile.

  3.  (AUTOFIXUP) Gather stale data dictionary statistics prior to
database
      upgrade in off-peak time using:

        EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;

      Dictionary statistics do not exist or are stale (not up-to-date).

      Dictionary statistics help the Oracle optimizer find efficient SQL
      execution plans and are essential for proper upgrade timing.
Oracle
      recommends gathering dictionary statistics in the last 24 hours
before
      database upgrade.

      For information on managing optimizer statistics, refer to the
12.2.0.1
      Oracle Database SQL Tuning Guide.

  4.  (AUTOFIXUP) Gather statistics on fixed objects prior the upgrade.

      None of the fixed object tables have had stats collected.

      Gathering statistics on fixed objects, if none have been gathered
yet, is
      recommended prior to upgrading.

      For information on managing optimizer statistics, refer to the
```

```
12.2.0.1
      Oracle Database SQL Tuning Guide.

  INFORMATION ONLY
  ================
  5.  To help you keep track of your tablespace allocations, the
following
      AUTOEXTEND tablespaces are expected to successfully EXTEND during
the
      upgrade process.

                                                  Min Size
      Tablespace                      Size        For Upgrade
      ----------                   ----------  -----------
      SYSAUX                          235 MB       500 MB
      SYSTEM                          210 MB       584 MB
      TEMP                             20 MB       150 MB
      UNDOTBS1                        210 MB       412 MB

      Minimum tablespace sizes for upgrade are estimates.

  6.  No action needed.

      Using default parallel upgrade options, this CDB with 1 PDBs will
first
      upgrade the CDB$ROOT, and then upgrade at most 1 PDBs at a time
using 2
      parallel processes per PDB.

      The number of PDBs upgraded in parallel and the number of parallel
      processes per PDB can be adjusted as described in Database
Upgrade Guide.

  ORACLE GENERATED FIXUP SCRIPT
  =============================
  All of the issues in database CDB2 container PDB$SEED
  which are identified above as BEFORE UPGRADE "(AUTOFIXUP)" can be
resolved by
  executing the following from within the container

    SQL>@/u01/app/oracle/cfgtoollogs/CDB2/preupgrade/
preupgrade_fixups.sql

=============
AFTER UPGRADE
=============

  REQUIRED ACTIONS
  ================
  None

  RECOMMENDED ACTIONS
  ===================
  7.  Upgrade the database time zone file using the DBMS_DST package.
```

```
      The database is using time zone file version 26 and the target
18.0.0.0.0
      release ships with time zone file version 31.

      Oracle recommends upgrading to the desired (latest) version of
the time
      zone file.  For more information, refer to "Upgrading the Time
Zone File
      and Timestamp with Time Zone Data" in the 18.0.0.0.0 Oracle
Database
      Globalization Support Guide.

  8.  (AUTOFIXUP) Gather dictionary statistics after the upgrade using
the
      command:

        EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;

      Oracle recommends gathering dictionary statistics after upgrade.

      Dictionary statistics provide essential information to the Oracle
      optimizer to help it find efficient SQL execution plans. After a
database
      upgrade, statistics need to be re-gathered as there can now be
tables
      that have significantly changed during the upgrade or new tables
that do
      not have statistics gathered yet.

  9.  Gather statistics on fixed objects after the upgrade and when
there is a
      representative workload on the system using the command:

        EXECUTE DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;

      This recommendation is given for all preupgrade runs.

      Fixed object statistics provide essential information to the
Oracle
      optimizer to help it find efficient SQL execution plans.  Those
      statistics are specific to the Oracle Database release that
generates
      them, and can be stale upon database upgrade.

      For information on managing optimizer statistics, refer to the
12.2.0.1
      Oracle Database SQL Tuning Guide.

  ORACLE GENERATED FIXUP SCRIPT
  ============================
  All of the issues in database CDB2 container PDB$SEED
  which are identified above as AFTER UPGRADE "(AUTOFIXUP)" can be
resolved by
  executing the following from within the container
```

```
    SQL>@/u01/app/oracle/cfgtoollogs/CDB2/preupgrade/
postupgrade_fixups.sql


==================
PREUPGRADE SUMMARY
==================
  /u01/app/oracle/cfgtoollogs/CDB2/preupgrade/preupgrade.log
  /u01/app/oracle/cfgtoollogs/CDB2/preupgrade/preupgrade_fixups.sql
  /u01/app/oracle/cfgtoollogs/CDB2/preupgrade/postupgrade_fixups.sql

Execute fixup scripts across the entire CDB:

Before upgrade:

1. Execute preupgrade fixups with the below command
$ORACLE_HOME/perl/bin/perl -I$ORACLE_HOME/perl/lib -I$ORACLE_HOME/
rdbms/admin $ORACLE_HOME/rdbms/admin/catcon.pl -l /u01/app/oracle/
cfgtoollogs/CDB2/preupgrade/ -b preup_CDB2 /u01/app/oracle/cfgtoollogs/
CDB2/preupgrade/preupgrade_fixups.sql

2. Review logs under /u01/app/oracle/cfgtoollogs/CDB2/preupgrade/

After the upgrade:

1. Execute postupgrade fixups with the below command
$ORACLE_HOME/perl/bin/perl -I$ORACLE_HOME/perl/lib -I$ORACLE_HOME/
rdbms/admin $ORACLE_HOME/rdbms/admin/catcon.pl -l /u01/app/oracle/
cfgtoollogs/CDB2/preupgrade/ -b postup_CDB2 /u01/app/oracle/cfgtoollogs/
CDB2/preupgrade/postupgrade_fixups.sql

2. Review logs under /u01/app/oracle/cfgtoollogs/CDB2/preupgrade/

Preupgrade complete: 2018-06-13T15:09:00
```

Pre-Upgrade Information Tool Warnings and Recommendations for Oracle Database

Analyze any Pre-Upgrade Information Tool warnings before you upgrade to the new release of Oracle Database. For each item that the tool reports, it provides you with information about how to fix the issue or warning.

For more detailed information , refer to My Oracle Support note 472937.1 for information about installed database components and schemas. Refer to My Oracle Support note 753041.1 for information about diagnosing components with `NON VALID` status.

- Updating Access Control Lists and Network Utility Packages
  Use this procedure to update access control lists (ACLs) and Network Utility Packages.

- Evaluate Dependencies and Add ACLs for Network Utility Packages
  You can receive a warning about network utility package dependencies. Use this procedure to evaluate the dependencies, and provide access by adding the appropriate access control lists (ACLs).
- About Database Links with Passwords from Earlier Oracle Database Releases
  This information is important only for downgrading to your original database release after performing the upgrade.
- About Oracle Database Warnings for TIMESTAMP WITH TIME ZONE Data Type
  Oracle Database upgrades include updated time zone data types, which may affect existing `TIMESTAMP WITH TIME ZONE` data types.

Related Topics

- https://support.oracle.com/rs?type=doc&id=472937.1
- https://support.oracle.com/rs?type=doc&id=753041.1

Updating Access Control Lists and Network Utility Packages

Use this procedure to update access control lists (ACLs) and Network Utility Packages.

Starting with Oracle Database 12*c*, the access control of the UTL packages is implemented using Oracle Database Real Application Security. UTL packages include UTL_TCP, UTL_SMTP, UTL_MAIL, UTL_HTTP, and UTL_INADDR. The access control does not require Oracle XML DB.

1. Ensure that the logged-in user has the `connect` privilege for the host and port specified by `DBMS_LDAP.init`. There is new behavior for the `DBMS_LDAP` PL/SQL package and the `HttpUriType` type. Because of this new behavior, you must create or update access control lists (ACLs) after you upgrade to the new Oracle Database release.

   For example, if your application depends on the `DBMS_LDAP` package, then the error "ORA-24247: network access denied by access control list (ACL)" can occur. To avoid this error, the logged-in user must have the `connect` privilege for the host and port specified by `DBMS_LDAP.init`.

2. If you have any of the following packages installed, then you can be required to reinstall these packages after upgrade:

   - `UTL_TCP`
   - `UTL_SMTP`
   - `UTL_MAIL`
   - `UTL_HTTP`
   - `UTL_INADDR`

   Ensure that you have the latest version of these packages for the new Oracle Database release.

> **See Also:**
>
> *Oracle Database Real Application Security Administrator's and Developer's Guide* for information about configuring access control lists

Evaluate Dependencies and Add ACLs for Network Utility Packages

You can receive a warning about network utility package dependencies. Use this procedure to evaluate the dependencies, and provide access by adding the appropriate access control lists (ACLs).

1. Run the Pre-Upgrade Information Tool.
2. Check the output from the Pre-Upgrade Information Tool (`preupgrade.jar`) for warning messages, such as the following example:

```
WARNING: --> Database contains schemas with objects dependent on network
packages.
.... Refer to the Database Upgrade Guide for instructions to configure
Network ACLs.
.... USER WKSYS has dependent objects.
.... USER SYSMAN has dependent objects.
.... USER FLOWS_010600 has dependent objects.
.
```

3. Query the view `DBA_DEPENDENCIES` to obtain more information about the dependencies. For example:

```
SELECT * FROM DBA_DEPENDENCIES
WHERE referenced_name IN
('UTL_TCP','UTL_SMTP','UTL_MAIL','UTL_HTTP','UTL_INADDR','DBMS_LDAP')
  AND owner NOT IN ('SYS','PUBLIC','ORDPLUGINS');
```

4. To ensure that the new access controls are part of your upgrade testing, prepare a post-upgrade script to make the scripts available in your database environment.

   Use the package `DBMS_NETWORK_ACL_ADMIN` to update your database access control lists (ACLs). You use this package to create, assign, and add privileges to the new access controls so that the updated access control packages can work as they did in prior releases. Refer to the example script provided in *Oracle Database Real Application Security Administrator's and Developer's Guide* to see how to use `DBMS_NETWORK_ACL_ADMIN` to update your access control list.

5. After the upgrade, grant specific required privileges. Access is based on the usage in the original database.

Related Topics

- *Oracle Database Real Application Security Administrator's and Developer's Guide*

About Database Links with Passwords from Earlier Oracle Database Releases

This information is important only for downgrading to your original database release after performing the upgrade.

During the upgrade to the new Oracle Database release, any passwords in database links are encrypted.

- To downgrade to the release from which you upgraded, you must drop all of the database links with encrypted passwords before the downgrade. Consequently, the database links are nonexistent in the downgraded database.
- If you must be able to downgrade to your original release, then save the information about affected database links from the `SYS.LINK$` table, so that you can recreate the database links after the downgrade.
- For information about earlier releases, refer to the original documentation for the Oracle Database release from which you upgraded. Also refer to your platform-specific *Oracle Database Installation Guide* for the earlier release.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for information about authentication and database links

About Oracle Database Warnings for TIMESTAMP WITH TIME ZONE Data Type

Oracle Database upgrades include updated time zone data types, which may affect existing `TIMESTAMP WITH TIME ZONE` data types.

The time zone files supplied with Oracle Database 12*c* are updated to reflect changes in transition rules for some time zone regions. The changes may affect existing `TIMESTAMP WITH TIME ZONE` data types.

Oracle recommends that you ensure that you have the latest time zone files before you upgrade the database. If the time zone file version of the database you are upgrading is not the most recent version of the time zone file available for the new release of Oracle Database, then the Pre-Upgrade Information Tool displays a warning and describes how to proceed. The following table describes the warnings and summarizes how to resolve a mismatch in time zone file versions.

> **Caution:**
>
> The `TIMESTAMP WITH TIME ZONE` data stored in the database can become corrupted during the upgrade if there is a time zone file version mismatch.

Table A-2 Choices for Fixing the Time Zone File Version

Time Zone Version On the Database Being Upgraded	When to Fix the Time Zone Files
Earlier than the most current version included in the new database release and the Pre-Upgrade Information Tool displays "Database is using a time zone file older than version *n*."	After completing the database upgrade. Use the `DBMS_DST` PL/SQL package and follow the instructions in "Steps to Upgrade Time Zone File and Timestamp with Time Zone Data. Refer to the following document: *Oracle Database Globalization Support Guide*.
Later than the version included in the new database release and the Pre-Upgrade Information Tool displays "Database is using a time zone file greater than version *n*."	Before beginning the database upgrade. You must patch the Oracle home by using an `RDBMS DST` patch with the appropriate patch for the time zone file version in use. Apply the patch for each database that you want to upgrade. Otherwise, the upgrade script terminates without upgrading the database. The `RDBMS DST` patches are available from My Oracle Support note ID 412160.1.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*
- https://support.oracle.com/rs?type=doc&id=1509653.1
- https://support.oracle.com/rs?type=doc&id=412160.1
- *Oracle Database Globalization Support Guide*

B

Upgrading with Oracle Database Upgrade Assistant (DBUA)

Database Upgrade Assistant (DBUA) provides a graphical user interface to guide you through the upgrade of Oracle Database. DBUA works for CDB and non-CDB database systems.

> **Note:**
>
> You can start DBUA in silent mode, which does not present a user interface. Silent mode can be useful for large roll-outs and scripts.

- Recommendations for Using DBUA
  Review this topic to use Database Upgrade Assistant (DBUA) for multitenant architecture.
- About Stopping DBUA When Upgrading
  You must complete an upgrade manually if you stop DBUA.
- How DBUA Processes the Upgrade for Oracle Database
  You can start DBUA as part of the database software installation, or you can start it manually after installing the software.
- Upgrade Scripts Started by DBUA
  During the upgrade, DBUA automatically runs the appropriate upgrade scripts to automate the upgrade and minimize downtime.
- Moving a Database from an Existing Oracle Home
  Use this procedure to migrate Oracle Databases to another Oracle Database 18c or later home.
- Using DBUA to Upgrade the Database on Linux, Unix, and Windows Systems
  To upgrade a database using the DBUA graphical user interface, perform these steps from within the new Oracle home.
- Using DBUA in Silent Mode to Upgrade Oracle Database
  You can DBUA with the `-silent` command line option to carry out noninteractive (“silent”) upgrades using DBUA.

Related Topics

- Using DBUA in Silent Mode to Upgrade Oracle Database
  You can DBUA with the `-silent` command line option to carry out noninteractive (“silent”) upgrades using DBUA.

Recommendations for Using DBUA

Review this topic to use Database Upgrade Assistant (DBUA) for multitenant architecture.

The preferred option for upgrading Oracle Database is to use the AutoUpgrade utility. However, you can use DBUA to upgrade multitenant architecture container databases (CDB), and pluggable databases (PDBs). Starting with Oracle Database 20c, you can no longer use DBCA to upgrade to a non-CDB Oracle Database.

> **Note:**
>
> Starting with Oracle Database 20c, non-CDB architecture is desupported. You must upgrade a non-CDB Oracle Database to a PDB on a CDB.

- Before using DBUA to upgrade your system, Oracle strongly recommends that you run the Pre-Upgrade Information Tool manually. DBUA runs the Pre-Upgrade Information Tool as part of the prerequisite checks it performs before starting the upgrade. However, to reduce downtime, Oracle recommends that you run the Pre-Upgrade Information Tool as part of your upgrade planning, so that you can analyze the database, and take proactive steps before your planned upgrade date.
- To use guaranteed restore points, ensure that the database ARCHIVE LOG and FLASHBACK modes are on during upgrade. You can confirm that they are on by entering the following SQL command:

  ```
  SQL> select log_mode,flashback_on from v$database;
  ```

- If the database instance is not running, then DBUA tries to start the instance. If the instance is up and running, then DBUA connects to it.
- If you restore your database manually (not using DBUA), then before starting DBUA, remove the `Welcome_`*`SID`*`.txt` file, which is located in the directory *`ORACLE_HOME`*`/cfgtoollogs/dbua/logs/`. If DBUA finds this file, then DBUA starts in a re-run operation.
- Restore scripts generally enable you to restore your database (pdb, high availability, or Oracle RAC) back to the earlier release and earlier Oracle home location. However, if you have registered your database with Oracle Internet Directory (OID), then the restore script cannot unregister Oracle Internet Directory. You must log in as an authorized user, and unregister the later release database manually.
- If Oracle Database Vault is enabled, then review in this document “Requirement for Upgrading Oracle Databases That Use Oracle Database Vault”.

About Stopping DBUA When Upgrading

You must complete an upgrade manually if you stop DBUA.

If you stop the upgrade, but do not restore the database, then you cannot continue to upgrade using DBUA. You must instead continue the upgrade using the manual (command line) upgrade procedure. You cannot go back to the original Oracle Database server unless you restore your database.

Related Topics

- Manually Upgrading a Multitenant Container Oracle Database (CDB)

How DBUA Processes the Upgrade for Oracle Database

You can start DBUA as part of the database software installation, or you can start it manually after installing the software.

If you install the new Oracle Database software, and you specify that you are upgrading an existing Oracle database, then DBUA starts automatically. You can also start DBUA independently after the installation is completed.

While the upgrade is in process, DBUA shows the upgrade progress for each component. DBUA writes detailed trace and log files and produces a complete HTML report for later reference. To enhance security, DBUA automatically locks new user accounts in the upgraded database. DBUA then proceeds to create new configuration files (parameter and listener files) in the new Oracle home.

DBUA does not begin the upgrade process until all the pre-upgrade steps are completed.

Related Topics

- Rerunning Upgrades for Oracle Database
- Tasks to Prepare for Oracle Database Upgrades

Upgrade Scripts Started by DBUA

During the upgrade, DBUA automatically runs the appropriate upgrade scripts to automate the upgrade and minimize downtime.

During the prerequisite phase, DBUA runs the Pre-Upgrade Information Tool script, and uses the following logic to modify or create new required tablespaces:

- If the data files are auto-extensible and have enough disk space to grow, then DBUA continues with the upgrade.
- If the data files are not auto-extensible, then DBUA prompts you and makes the files auto-extensible.
- If the tablespaces are auto-extensible and the `MAXSIZE` initialization parameter needs adjustment, then DBUA prompts you to for this adjustment, and adjusts the `MAXSIZE` parameter.
- If there is not enough disk space to grow, then DBUA prompts you to create space by adding more data files. DBUA does not automatically add new data files, because DBUA cannot determine where to create the files.

DBUA addresses many issues found during the prerequisite phase. For example, DBUA can ensure that the correct time zone file is used, and make ACL adjustments for network access control lists.

During the upgrade phase, DBUA runs `catctl.pl`, which runs the upgrade processes in parallel instead of serially. Parallel runs optimize utilization of CPU resources to hasten the upgrade and minimize downtime.

Moving a Database from an Existing Oracle Home

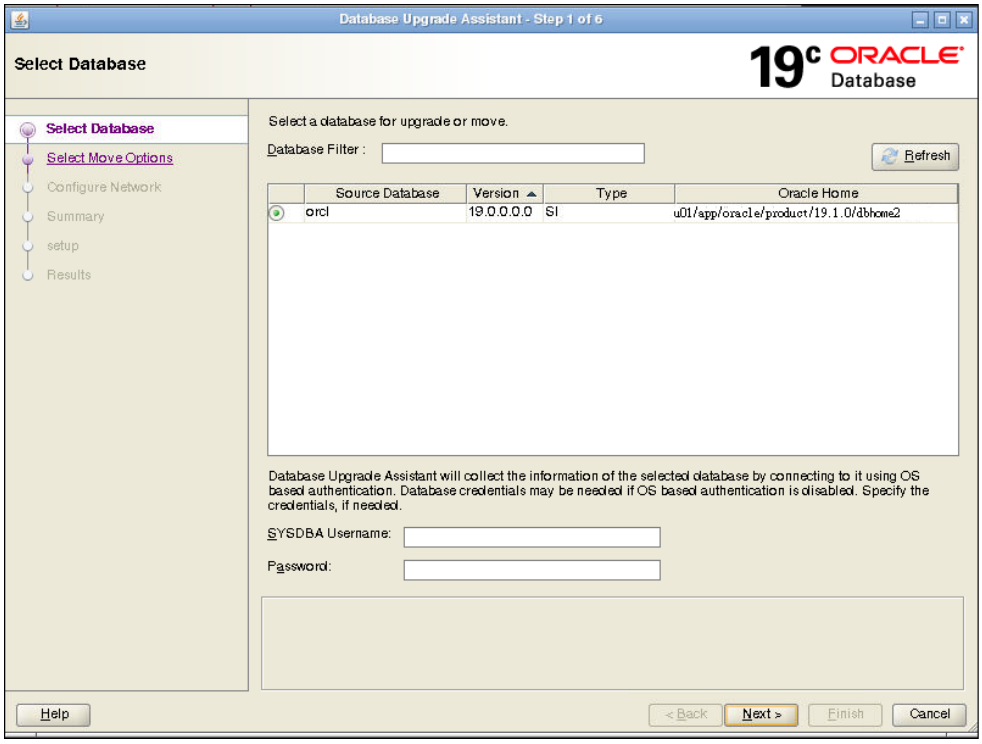
Use this procedure to migrate Oracle Databases to another Oracle Database 18c or later home.

With Oracle Database 18c and later releases, you can use Database Upgrade Assistant (DBUA) to migrate Oracle Database databases from an existing Oracle home to another Oracle home.

1. Start DBUA.

   DBUA opens the Select Database window.

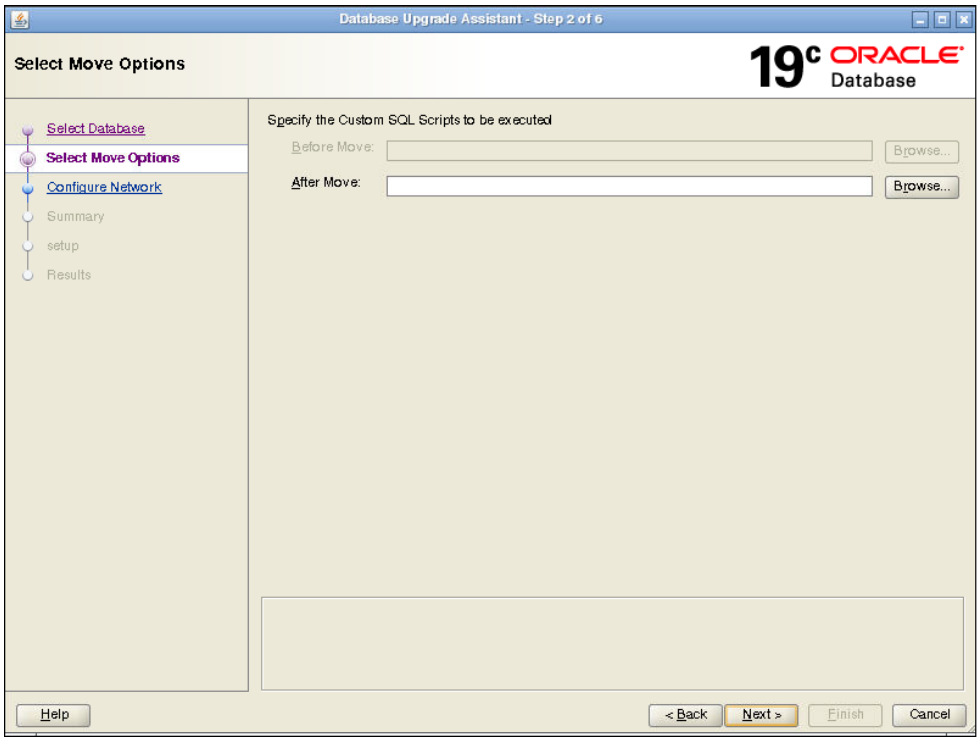
   All databases on the server are listed. DBUA indicates the type of operation that you can perform for each database (upgrade, move, in place), depending on the database release and location.



   Select a database that you want to move to the new Oracle home. If you have not enabled operating system authentication for the database, then provide the SYSDBA user name and password for the database that you select.

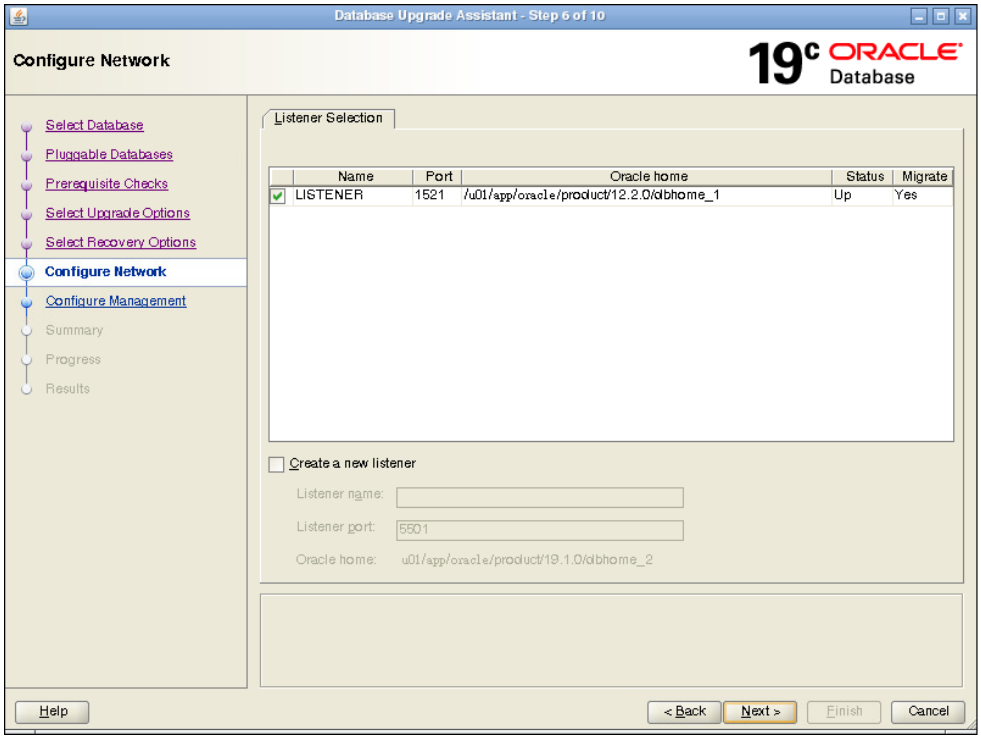
   Click **Next**. The Move Database Options window appears.

2. In the Select Move Options window, you can specify custom SQL scripts that you want to run after moving the database, and identify where the files are located.



Click **Next**. The Configure Network window appears.

3. On single-instance systems, you can either select an existing listener, or create a new listener. If you create a new listener, then you must provide a listener name, and a port number for the listener.



Click **Next**. The Database Move Summary window appears.

4. Review the summary for the move operation and click **Next**.
5. The Setup window appears, which shows DBUA processes as it moves the database.
6. When the move operation completes, click **Finish**.
7. The Results window appears. You can review the source and target database information, and check move steps, or check log files. When you are finished with your review, click **Close**.

Using DBUA to Upgrade the Database on Linux, Unix, and Windows Systems

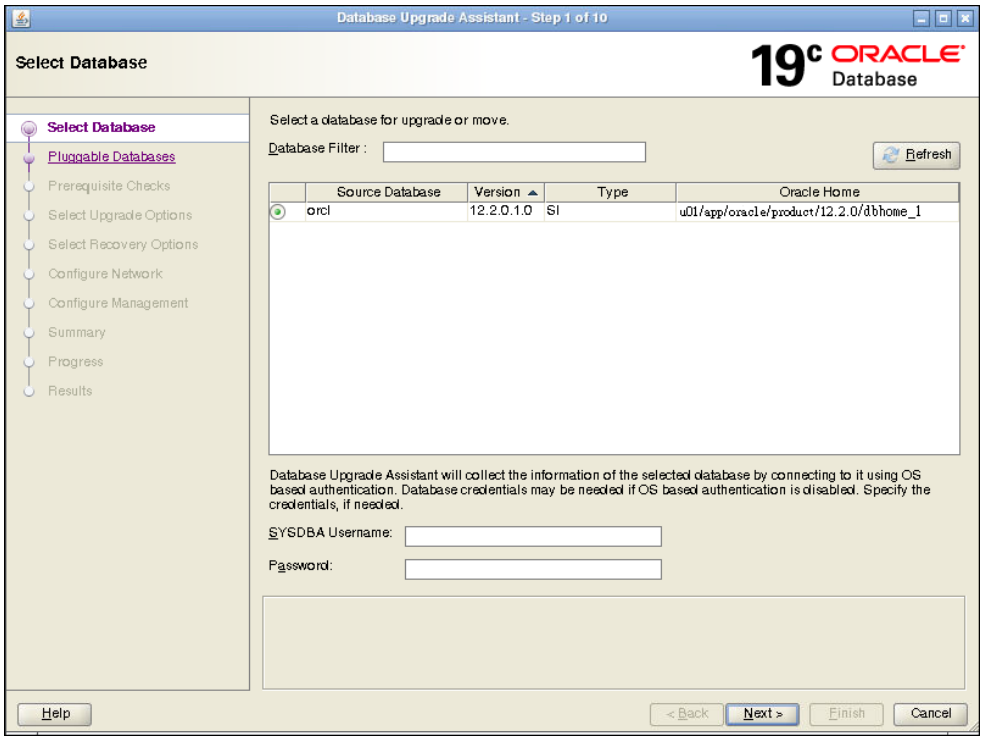
To upgrade a database using the DBUA graphical user interface, perform these steps from within the new Oracle home.

On Microsoft Windows systems (Windows), run DBUA either as an Oracle Database administrative user (a user with the operating system-assigned `ORA_DBA` role), or as the Oracle installation owner account.

1. Start Oracle Database Upgrade Assistant (DBUA) from the Oracle home where the new database software is installed. The `dbua` executable is located in the directory path *`ORACLE_HOME`*`/bin`.
   - On Linux or Unix platforms, log in as a user with SYSDBA privileges, and enter the following command at a system prompt in the new home for Oracle Database 19c:

     ```
     ./dbua
     ```

   - On Windows operating systems, select **Start**, then **Programs**, then **Oracle *HOME_NAME***, then **Configuration and Migration Tools**, and then **Database Upgrade Assistant**.
2. The Select Database window displays. If you have earlier release Oracle Database installations, then these installations are listed as available to upgrade.



If you need help on any DBUA window, or if you want to consult more documentation about DBUA, then click **Help** to open the online help.

If needed, enter the SYSDBA user name and password for the database that you select.

If you run DBUA from a user account that does not have `SYSDBA` privileges, or if the source database does not have operating system authentication, then you must enter the user name and password credentials to enable `SYSDBA` privileges for the selected database. If the user account you use has SYSDBA privileges, or you used operating system authentication, then you do not need to enter the user name and password.

Click **Next** after making your selection.

Note:

- You can select only one database at a time.
- With single-instance upgrades, if the database does not appear in the list, then check to see if an entry with the database name exists in `/etc/oratab`. If the database is not listed there, then direct DBUA to upgrade particular databases:
  - If your single-instance database is not listed in `/etc/oratab`, and DBUA can connect to the database, then you can direct DBUA to upgrade that database explicitly by starting DBUA using the command-line arguments `-sid` *`Oracle_SID`*, `-oracleHome` *`Oracle_home`*, and `sysDBAPassword` *`password`* as a command-line argument. For example:

    ```
    dbua -sid Oracle_SID –oracleHome /u01/app/oracle/
    18.1.0/dbhome1 -sysDBAUserName SYS -sysDBAPassword
    password
    ```

- If your account does not have SYSDBA privileges, or you do not have operating system authentication set up, then you can use the following syntax to connect, where *`mydb`* is your Oracle Database SID, *`username`* is a user name with SYSDBA privileges, and *`password`* is that user name's password:

  ```
  dbua -sid mydb –oracleHome /u01/app/oracle/18.1.0/
  dbhome1 -sysDBAUserName - username -sysDBAPassword -
  password
  ```

- Oracle Real Application Clusters (Oracle RAC) upgrades: If the database does not appear on the list, then enter the following `crsctl` command to check for Oracle RAC instances:

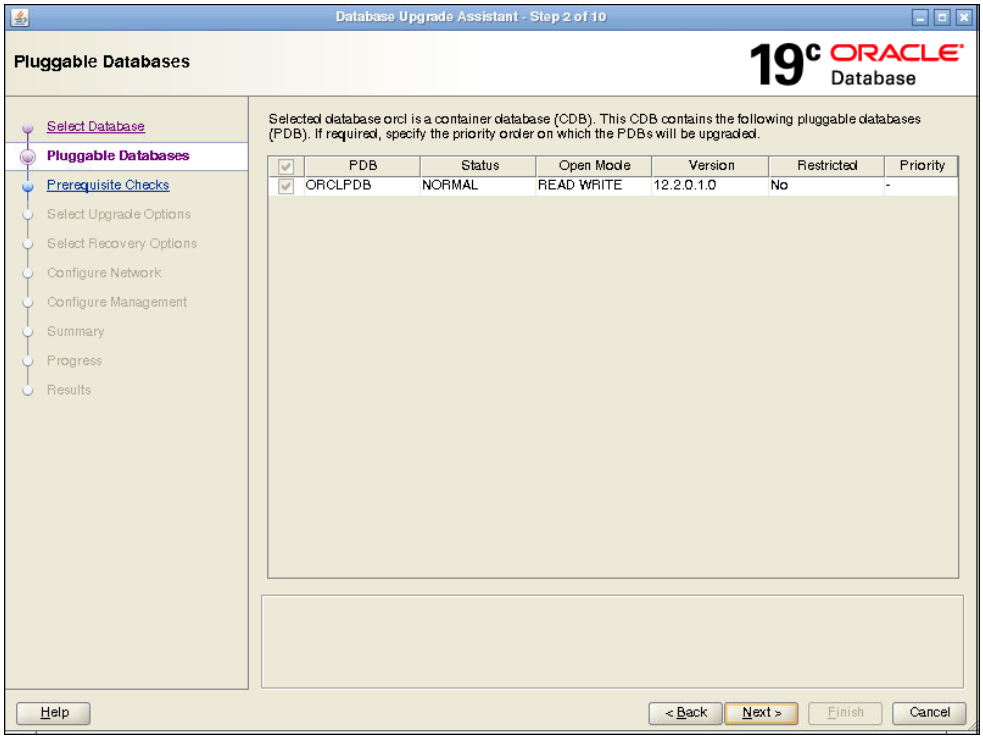
  ```
  crsctl status resource -t
  ```

  You can also enter the following command to check for a particular Oracle RAC database, where *`db_name`* is the Oracle RAC database name:

  ```
  crsctl status resource ora.db_name.db
  ```

- On Microsoft Windows, the following security changes affect authentication and user accounts:
  - For security reasons, Windows NTS authentication using the NTLM protocol is no longer supported. Kerberos authentication is the only supported authentication. In this release, NTS does not work either in Windows NT domains, or in domains with Windows NT controllers.
  - Oracle uses standard Microsoft Windows user accounts instead of the Windows `LocalSystem` account to run Oracle database services. Reducing the account access privileges for the Oracle installation owner provides better security on Microsoft Windows.

3. If the selected database is a multitenant container database (CDB), then DBUA displays the Pluggable Databases window. The Pluggable Databases window lists the pluggable databases contained in the CDB. The listed PDBs are upgraded as part of the upgrade for the selected CDB.

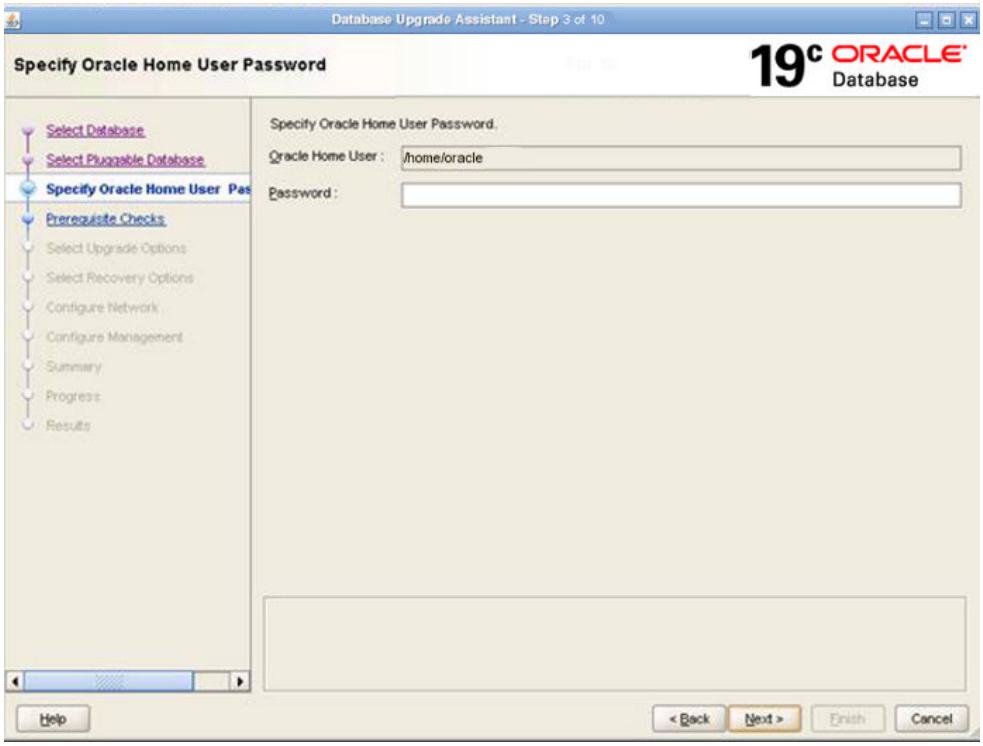


You can select the upgrade priority for PDBs. Click in the priority column for each PDB, and enter a numeric value for upgrade priority, where 1 is upgraded first, 2 is upgraded second, and so on.

By default, `CDB$ROOT`, `PDB$SEED`, and all PDBs that are plugged into the CDB are upgraded. If you do not want some PDBs to be upgraded now, then unplug those PDBs.

When you have completed selecting PDBs and upgrade priorities, click **Next**.

4. Windows platforms only: If the upgrade target home is a secure home that is associated with an Oracle home user, then the Specify Oracle Home User Password window opens. For other platforms, proceed to the next step.



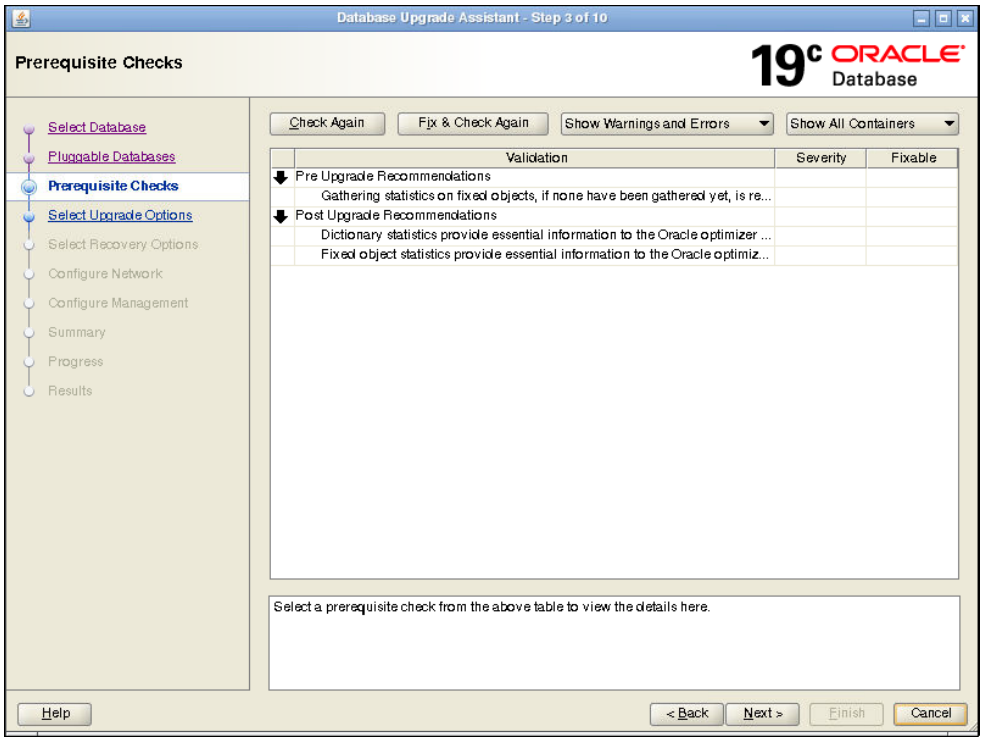
Provide the Oracle home user name, and provide the password for this user account, and click **Next**.

5. The Prerequisite Checks window opens. DBUA analyzes the databases, performing preupgrade checks and displaying warnings as necessary. The following is a list of examples of DBUA checks, and of actions that DBUA performs on the database:

   - Empty database recycle bin.
   - Identify invalid objects.
   - Identify deprecated and desupported initialization parameters.
   - Identify time zone data file version.

   The analysis takes several minutes to complete.

   When DBUA finishes its analysis, the Prerequisite Checks window displays again, showing the results of the checks.



The Prerequisite Checks window shows the checks that DBUA has completed, and the severity of any errors discovered. When DBUA finds errors, it indicates which errors are fixable, and what action you can take to correct the error.

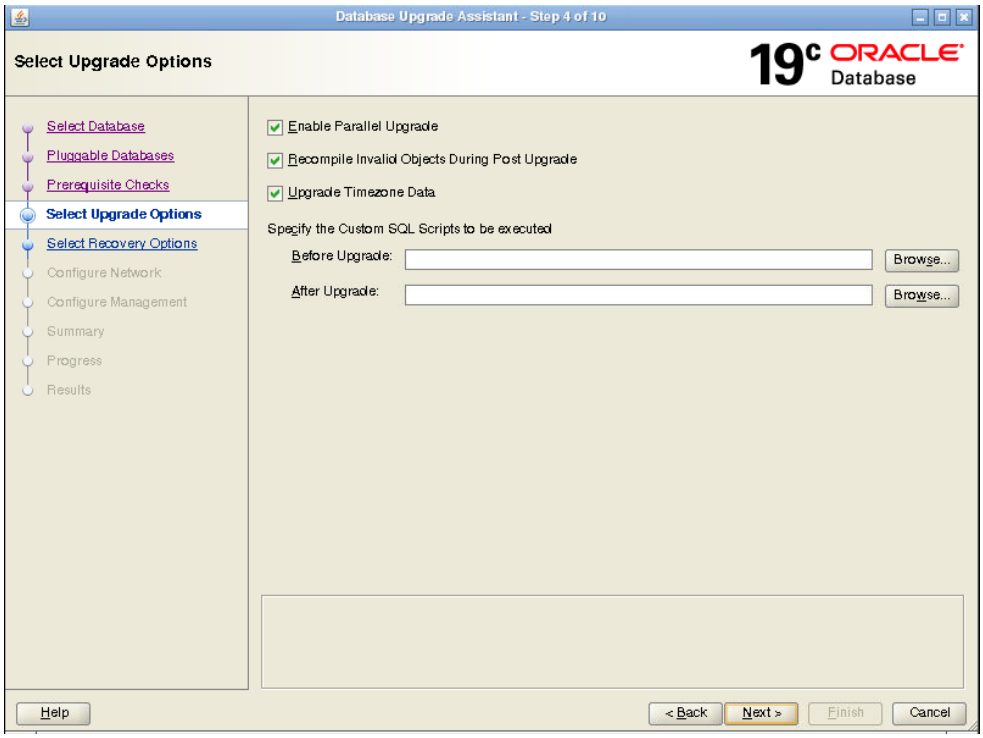
Select **Fix & Check Again** if any errors that DBUA can fix appear.

If DBUA detects errors that it cannot correct, then fix the cause of the error manually, and select **Check Again**.

If DBUA finds no errors or warnings, then the DBUA automatically bypasses this window and proceeds to the next window.

When you have fixed detected errors, click **Next**.

6. The Select Upgrade Options window displays.



This window provides the following options:

Enable Parallel Upgrade

Select this option if you want to enable parallelism during the upgrade process. Running upgrade processes in parallel reduces the time required to perform the upgrade, based on the number of CPUs available to handle the running of scripts and processes simultaneously.

Recompile Invalid Objects During Post Upgrade

This option recompiles all invalid PL/SQL modules after the upgrade is complete. If you do not have DBUA recompile invalid objects in its post-upgrade phase, then you must manually recompile invalid objects after the database is upgraded.

Upgrade Time Zone Data

This option updates the time zone data file for this release. If you do not select this option, then you must update the time zone configuration file manually after the upgrade.

Specify custom SQL scripts to be executed.

If you want to run custom SQL scripts as part of your upgrade, then select this option. As needed, click **Browse** for the **Before Upgrade** or **After Upgrade** input fields. Navigate to the location where your custom SQL scripts are located.

When you have made your selections, click **Next**.

7. The Select Recovery Options window appears. To recover the database if a failure occurs during upgrade, select from one of the following options:

   - **Use Flashback and Guaranteed Restore Point**.

You can create a new Guaranteed Restore Point, or use an existing one. If you use an existing restore point, then click the selection field to select the restore point that you want to use.

> **Note:**
>
> If the database that you are upgrading has Oracle Data Guard physical standbys, then you must first create a guaranteed restore point on each standby before you create one on the primary database. If you do not create restore points on each standby first, then you must recreate all standby databases again after using the guaranteed restore point to downgrade the primary database. After the upgrade is successful, you must manually drop all guaranteed restore points on the standbys.

- **Use RMAN Backup**

  Select among the following RMAN backup options:

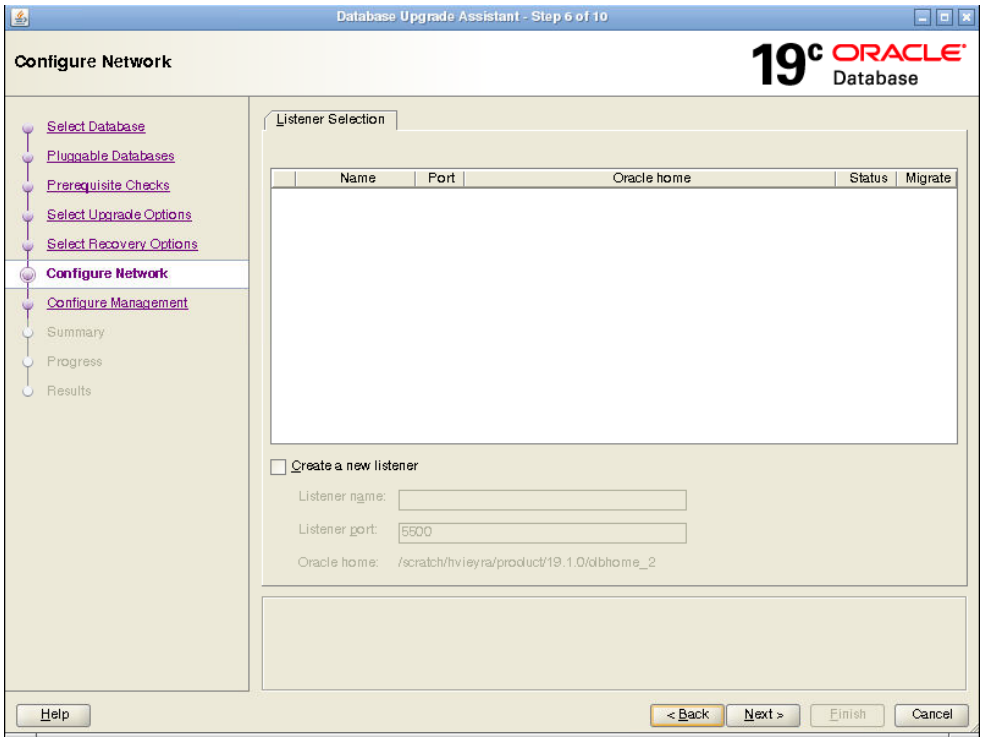
  - **Create a new Offline RMAN backup**. If you select this option, then provide a path where you want to place the backup.
  - **Create a New Partial Offline RMAN Backup with User Tablespace**. If you select this option, then user tablespaces are placed into read-only mode during the upgrade.
  - **Use the Latest Available Full RMAN Backup.** If you select this option, then click **View/Edit Restore Script** to select the backup that you want to use.

- **I have my own backup and restore strategy**.

  Select this option only if you have a third-party backup solution in place for your existing database.

When you have made your selections, click **Next**.

8. For single-instance database installations, the Configure Network window opens. Select one or more listeners from the source Oracle home that you want to migrate to the new upgraded Oracle home, or create a new listener during installation.



The Listener Selection area of the Network Configuration window shows a table with the following columns:

- Select column. Select the listeners that you want to update.
- **Name** This column shows listener names.
- **Port** This column shows the ports on which listeners are configured.
- **Oracle Home** This column shows the Oracle home where listeners are configured.
- **Status** This column shows the listener status (up or down).
- **Migrate** Select this column, and choose `Yes` to migrate, or `No` if you do not want to migrate.

You can also select to create a new listener. If you create a new listener, then provide the listener name, the Oracle home where you want it placed, and the port that you want to configure the listener to monitor.

After you make your choices, DBUA completes the following steps for any listeners that you migrate:

a. DBUA adds the selected listener to the `listener.ora` file of the target Oracle home, and starts it.

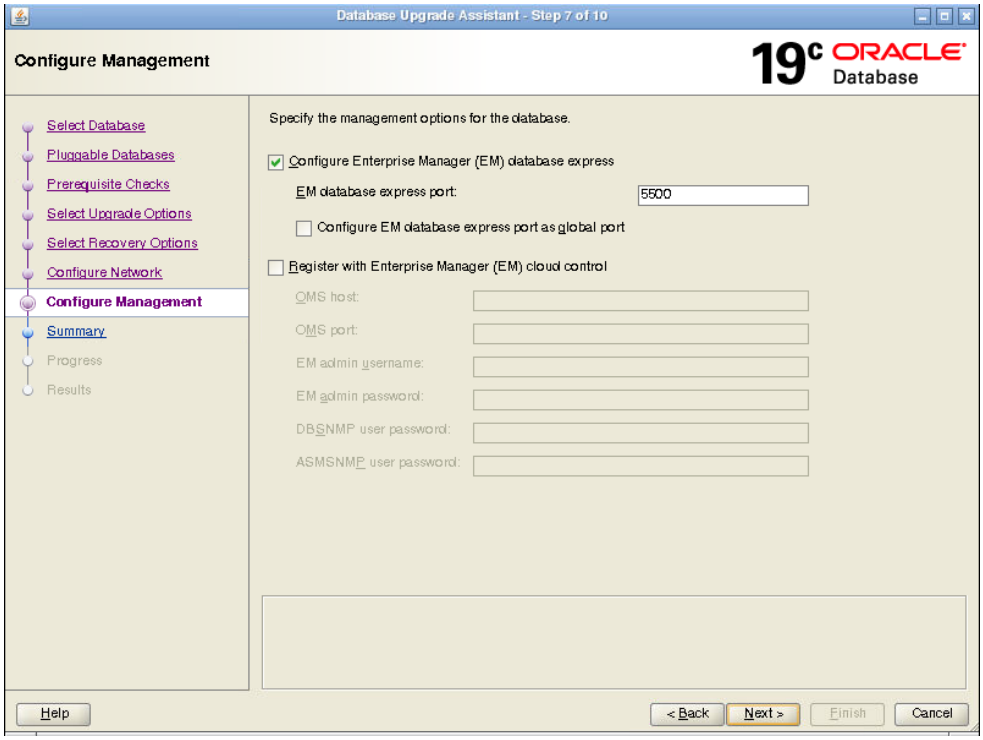
b. DBUA removes the entry of the upgraded database from the old (source) `listener.ora` file.

c. DBUA reloads the `listener.ora` file in both the source and target Oracle Database environments.

> **Note:**
>
> If there are other databases registered with the same listener, then their new client connection requests can be affected during listener migration.

Click **Next** when you have completed your choices.

9. The Configure Management window appears. In the Configure Management window, select the management options:



- **Configure Enterprise Manager (EM) database express**

  Oracle Enterprise Manager Database Express is a web-based database management application that is built into Oracle Database. EM Express replaces the DB Control component that was available in earlier releases. If you select to configure Enterprise Manager Database Express, then Enter the EM Database Express Port number. For example: 5500. You can also select the check box to configure the express port as the global port.

- **Register with Enterprise Manager (EM) Cloud Control**

  Registering with Oracle Enterprise Manager Cloud Control adds the database and its related entities, such as Listener, Oracle ASM disk groups, and Oracle Clusterware, as targets that you can manage with EM Cloud Control.

  If you select this option, then you must provide information in the following fields:

  – OMS Host

- OMS Port
- EM Admin Username
- EM Admin Password
- DBSNMP User Password
- ASMSNMP User Password

When you have completed entering information, click **Next**.

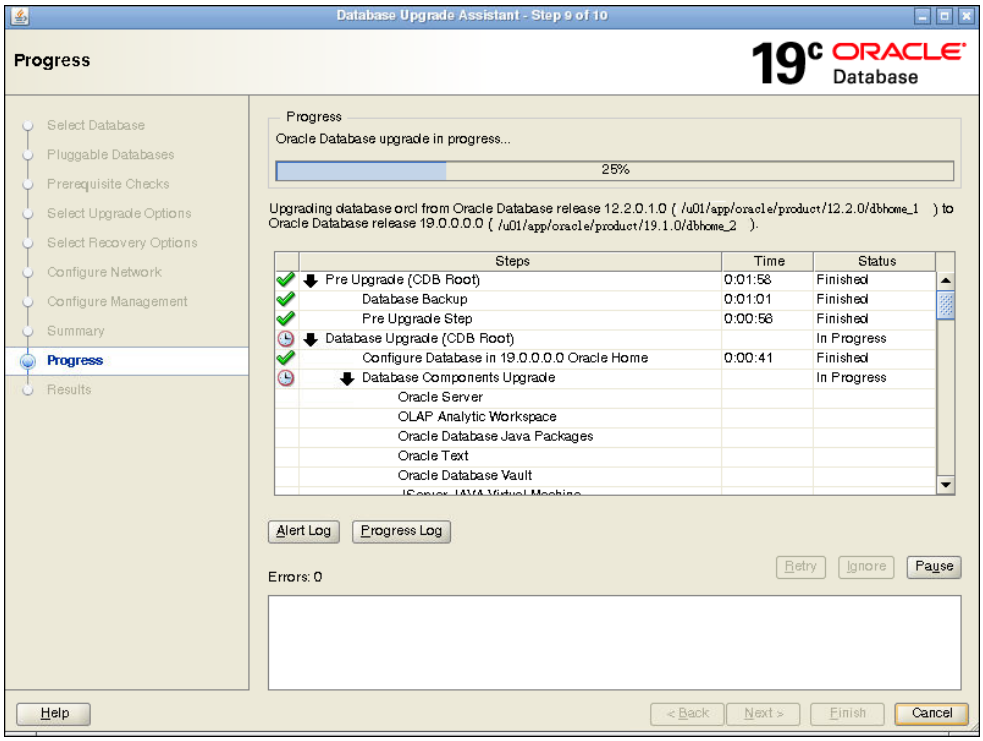
10. The Summary window opens. The Summary window shows the information that you have provided for the upgrade. Scroll down the list to review the information. The summary includes information such as the following:

    - Source Database
    - Target Database
    - Pluggable Databases
    - Pre-Upgrade Checks
    - Initialization Parameters changes
    - Timezone Upgrade

    Check your selections. Then, either select a link to the item that you want to change, or click **Back** to go to earlier pages, or select **Finish**:

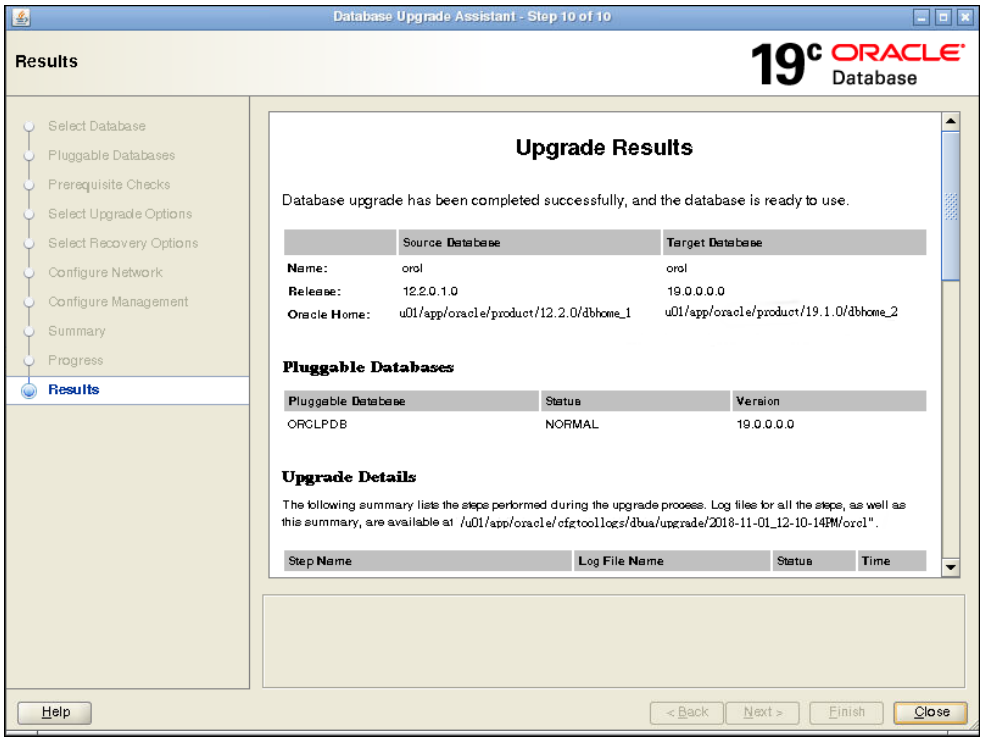
    - If you see information in the Summary window that you want to correct, then click a link on an item that you want to update, or click **Back** to navigate backward through the DBUA configuration interview.
    - Click **Finish** if the information that you see in the Summary window is correct. The upgrade begins after you select **Finish**.

    The Progress window displays with the progress bar, as DBUA begins the upgrade. The Progress window displays a table that shows the steps DBUA is completing during the upgrade. This table shows the time duration, and the upgrade steps status as the upgrade proceeds. DBUA provides a **Stop** button in case you must cancel the upgrade at this point.



When the upgrade has progressed through finishing the upgrade of the CDB root and each PDB seed, the Progress window marks the status as **Finished**.

11. After the upgrade is complete, the Results window opens. The Results window displays information about the original database, and about the upgraded database. The Upgrade Results report also shows changes that DBUA made to the initialization parameters. If you are upgrading a multitenant architecture database, then the Results window also shows pluggable databases, and the directory where log files are stored after the upgrade. Scroll down to see more details about preupgrade checks. If the upgrade is successful, then the Upgrade Results field reports the results, and you do not see warning messages. If the upgrade was unsuccessful, as this example image shows, then the **Restore Database** button is displayed on the lower right corner below the display field. You can click this button to start a database restoration.



12. Optional: Examine the log files to obtain more details about the upgrade process. If the Oracle base environment variable is set, then the DBUA log files are located in the path */ORACLE_BASE*`/cfgtoollogs/dbua/upgradesession_`*timestamp*`/`*SID*. If Oracle base is not set, then the DBUA log files are located in the path */ORACLE_HOME*`/cfgtoollogs/dbua/upgradesession_`*timestamp*`/`*SID*

> **Note:**
>
> An HTML version of the Upgrade Results window is also saved in the log files directory. You can click the links in this HTML window to view the log windows in your browser.

If you are satisfied with the upgrade results, then click **Close** to quit DBUA.

13. After your upgrade is completed, carry out post-upgrade procedures described in this book. When you have completed post-upgrade procedures, your upgraded database is ready to use.

> **Caution:**
>
> To prevent unauthorized use of the database, Oracle recommends that you change all user passwords immediately after you upgrade your database.
>
> If the default security settings for Oracle Database 12c and later releases are in place, then passwords must be at least eight characters. Passwords such as `welcome` and `oracle` are not allowed.

Using DBUA in Silent Mode to Upgrade Oracle Database

You can DBUA with the `-silent` command line option to carry out noninteractive (“silent”) upgrades using DBUA.

. In silent mode, DBUA does not present a user interface. DBUA writes messages (including information, errors, and warnings) to a log file in *`ORACLE_HOME`*`/cfgtools/dbua/upgrade`*`session_timestamp`*, where *`session_timestamp`* represents the timestamp for the upgrade that DBUA has run. Oracle strongly recommends that you read the resulting DBUA log files to ensure a successful upgrade.

- Running DBUA in Silent Mode
  Use this procedure to start DBUA in noninteractive (or “silent”) mode.
- DBUA Command-Line Syntax for Active and Silent Mode
  Use this syntax to run Database Upgrade Assistant (DBUA) either interactively, or by using the `-silent` option.

Running DBUA in Silent Mode

Use this procedure to start DBUA in noninteractive (or “silent”) mode.

1. To start DBUA in silent mode, enter the `dbua -silent -sid` command. The command starts DBUA in silent mode, and identifies the database that you want to upgrade.

   For example, enter the following command, where the database name is ORCL:

   ```
   dbua -silent -sid ORCL &
   ```

DBUA Command-Line Syntax for Active and Silent Mode

Use this syntax to run Database Upgrade Assistant (DBUA) either interactively, or by using the `-silent` option.

Purpose

When you run DBUA by using the command-line option, you can specify all valid DBUA options in a script form. The script form enables you to avoid entering configuration information in a graphic user interface dialog.

File Path

`$ORACLE_HOME``/directory_name`

Syntax

Usage: dbua [*flag*] [*option*]

Flags

`-createPartialBackup` Flag to create a new offline partial RMAN backup by setting the user tablespaces in Read-Only mode.
`-backupLocation` Flag to identify the path for the RMAN backup.

`-disableParallelUpgrade` Flag to disable the parallel execution of database upgrade.
`-executePreReqs` Flag to run the preupgrade checks alone for the specified database.
`-sid | -dbName`
`-sid`
`-dbName`
`-help` - Shows this usage help.
`-ignorePreReqs` Ignore error conditions in preupgrade checks.
`-silent` Runs configuration in silent mode.
`-sid | -dbName`
`-sid`
`-dbName`
`-skipListenersMigration` Flag to bypass the listener migration process as part of the database upgrade.

Options

The DBUA command line options are as follows:

`[-asmsnmpPassword - ASMSNMP user password]`
`[-backupLocation - Directory-where-you-want-to-back-up-your-database-before-starting-upgrade]`
`[-createGRP - [True | false]` Create a guaranteed restore point when database is in archive log and flashback mode.
`[-createListener - [true [listenrName:lsnrPort] | false]]` Create a listener in later release Oracle home. If true, then add: *`listenerName:lsnrPort`*
`[-dbName - database-name]`
`[-oracleHome - Oracle-home-path-of-database]`
`[-sysDBAUserName - User-name-with-SYSDBA-privileges]`
`[-sysDBAPassword - Password-for-sysDBAUserName-user-name]`
`[-dbsnmpPassword - DBSNMP-user-password]`
`[-disableUpgradeScriptLogging - [true | false]]` This command disables the detailed log generation for running SQL scripts during the upgrade process. By default this is enabled. To enable the log generation, don't specify this command.
`[-emConfiguration - [DBEXPRESS | CENTRAL | BOTH | NONE]` Specifies Enterprise Manager information.
`[-dbsnmpPassword - DBSNMP-user password]`
`[-emPassword - Enterprise-Manager-administration-user-password]`
`[-emUser - Enterprise-Manager-Administration-username-to-add-or-modify-targets]`
`[-emExpressPort - port-for-EM-Express]`
`[-omsHost - Enterprise-Manager-management-server-host-name]`
`[-omsPort - Enterprise-Manager-management-server-port-number]`
`[-asmsnmpPassword - ASMSNMP-user-password]`
`[-ignoreScriptErrors - [true | false]]` Specify this flag for ignoring ORA errors during custom scripts.
`[-initParam - name=value,name=value,...]` Specify a comma separated list of initialization parameter values of the
format *name=value*,*name=value*
`[-initParamsEscapeChar - [escape character]` Escape character for comma when a specific initParam has multiple values. If the escape character is not specified, then backslash (\) is the default escape
`[-excludeInitParams - comma-delimited-list-of-initialization-parameters-to-exclude]`
`[-keepDeprecatedParams - [true | false]]` Retain (or not) deprecated parameters

during database upgrade.
`[-localListenerWithoutAlias]` Sets LOCAL_LISTENER without TNS Alias.
`[-listeners listenerName:Oracle Home, listenerName:Oracle Home, …]` Registers the database with existing listeners. Specify listeners by comma-delimited list in the form *listenerName*:*Oracle Home*. Listeners from earlier release Oracle homes are migrated to *newer release,lsnrName2* or -listeners *lsnrName1*:*Oracle home path*,-listeners *lsnrName2*:*Oracle home path*. DBUA searches specified listeners from the Grid Infrastructure home (if configured), the target home, and source home.
`[-localRacSid - local-System-Identifier-of-cluster-database]` Use if if the cluster database is not registered in the Oracle Cluster Registry (OCR).
`[-logDir - Path-to-a-custom-log-directory]`
`[-newGlobalDbName - New Global Database Name]` This option can only be used for Oracle Express Edition upgrades
`[-newSid - New System Identifier]` This option can only be used for Oracle Express Edition upgrades
`[-newInitParam - name=value,name=value,...]` Specify a comma-delimited list of initialization parameter values of the format `name=value,name=value`. Use this option to specify parameters that are allowed only on the target Oracle home.
`[-initParamsEscapeChar - escape character]` Specify an escape character for comma when a specific initParam has multiple values. If the escape character is not specified, then backslash (\) is the default escape.
`[-oracleHomeUserPassword - Oracle-Home-user-password]`
`[-pdbs - [All | NONE | pdb, pdb,…]]` Specify `ALL` to select all PDBs, `NONE` to select no PDBs, or provide a comma-delimited list with
the names of the pluggable databases (PDBs) that you want upgraded.
`-sid | -dbName`
`-sid - System Identifier`
`[-oracleHome - Oracle home path of the database]`
`[-sysDBAUserName - User-name-with-SYSDBA-privileges]`
`[-sysDBAPassword - Password-for-sysDBAUserName-user-name]`
`-dbName - Database-Name`
`[-oracleHome - Oracle-home-path-of-database]`
`[-sysDBAUserName - User-name-with-SYSDBA-privileges]`
`[-sysDBAPassword - Password-for-sysDBAUserName-user-name]`
`[-pdbsWithPriority - pdb:priority, pdb:priority, pdb:priority,...]` Specify a comma-delimited list of pluggable databases (PDB) that you want upgraded, including their corresponding priorities (1 the top priority).
`-sid | -dbName`
`-sid - system-identifier`
`[-oracleHome - Oracle-home-path-of-database`
`[-sysDBAUserName - User-name-with-SYSDBA-privileges]`
`[-sysDBAPassword - Password-for-sysDBAUserName-user-name]`
`-dbName - database-name`
`[-oracleHome - Oracle-home-path-of-the-database]`
`[-sysDBAUserName - User-name-with-SYSDBA-privileges]`
`[-sysDBAPassword - Password-for-sysDBAUserName-user-name]`
`[-performFixUp - [true | false]` Enable or disable fixups for the silent upgrade mode.
`[-postUpgradeScripts - SQLscript, SQLscript,…]` Specify a comma-delimited list of SQL scripts with their complete pathnames. After the upgrade completes, these SQL scripts are run.
`[-preUpgradeScripts - SQLscript, SQLscript,…]`Specify a comma-delimited list of SQL scripts with their complete pathnames. Before the upgrade starts, these SQL scripts are run.

`[-recompile_invalid_objects - [true | false]]` If true, then recompiles invalid objects as part of the upgrade.
`[-upgrade_parallelism - number]` Numeric value for the number of CPUs that you want to use for parallel upgrade.
`[-upgradeTimezone - [true | false]]` If true, then upgrades the timezone files of the database during the upgrade.
`[-upgradeXML - Path-to-existing-preupgrade-XML-file]` This option only applies to in-place database upgrades.
`[-useExistingBackup - [true | false]]` Use to enable restoration of the database using existing RMAN backup.
`[-useGRP - Name-of-existing-guaranteed-restore-point]` Use to enable restoration of the database using a specified guaranteed restore point.

Example B-1 Selecting a Database for Upgrade with DBUA

The following command selects the database `orcl` for upgrade:

```
dbua -sid orcl
```

> **Note:**
>
> You can use DBUA commands to set passwords. If the default Oracle Database security settings are in place, then passwords must be at least eight characters, and passwords such as `welcome` and `oracle` are not allowed.

Example B-2 Selecting a Database for Upgrade with DBUA Using Noninteractive ("Silent") Option

The following command selects the database `orcl` for upgrade using the noninteractive ("silent") option:

```
dbua -silent -sid orcl
```

Example B-3 Use Cases for Running DBUA in Noninteractive (”Silent") Mode

The examples that follow illustrate how you can use DBUA with the noninteractive (”silent") option to carry out a variety of upgrade scenarios.

```
dbua -silent -sid sidb112 -backupLocation /u01/sidb1123/backup -
sysDBAUserName sys -sysDBAPassword r3aDy2upg -oracleHome /u01/app/
product/11.2.0/dbhome_1 -upgradeTimezone true dbua -silent -sid
sidb1123 -backupLocation /u01/sidb1123/backup -sysDBAUserName sys -
```

```
sysDBAPassword r3aDy2upg -oracleHome /u01/app/product/11.2.0/dbhome_1 -
upgrade_parallelism 1 -upgradeTimezone true


dbua -silent -sid db1124 -backupLocation /u01/sidb1123/backup -
sysDBAUserName sys -sysDBAPassword r3aDy2upg -performFixUp true -
upgradeTimezone true


dbua -silent -dbName rdbcdb -oracleHome /u01/app/product/11.2.0/
dbhome_1 -sysDBAUserName sys -sysDBAPassword r3aDy2upg -
backupLocation /u01/sidb1123/backup -recompile_invalid_objects true -
upgradeTimezone true


dbua -silent -dbName amdb -oracleHome /u01/app/product/11.2.0/dbhome_1 -
sysDBAUserName sys -sysDBAPassword r3aDy2upg -recompile_invalid_objects
true -useGRP GRP_20170620bfupgrade -upgradeTimezone true


dbua -silent -dbName rdb12 -oracleHome /u01/app/product/12.2.0/dbhome_2
-sysDBAUserName sys -sysDBAPassword r3aDy2upg -backupLocation /u01/
sidb12/backup -recompile_invalid_objects true -upgradeTimezone true


dbua -silent -dbName ronedb -oracleHome /u01/app/product/12.2.0/
dbhome_2 -sysDBAUserName sys -sysDBAPassword r3aDy2upg -
recompile_invalid_objects true -upgradeTimezone true -createGRP true
```

> **Note:**
>
> Refer to *Oracle Database Security Guide* for information about security best practices.

Related Topics

- *Oracle Database Security Guide*

Index

Symbols

.NET
  PromotableTransaction deprecated, *10-49*

A

access control lists (ACLs), *10-39*
  granting access to network utility packages, *A-22*
  XDB ACLs migrated, *10-39*
ACFS-9427: Failed to unload ADVM/ACFS drivers, *5-8*
ACFS-9428 Failed to load ADVM/ACFS drivers, *5-8*
ACLs
  See access control lists (ACLs)
adding ACLs for network utility packages, *A-23*
addnode.bat
  deprecated, *10-8*
addnode.sh
  deprecated, *10-8*
adjusting after manual upgrades, *6-45*
amd_exists, *4-50*
apex_upgrade_msg, *4-50*
application code
  not changing after upgrade, *7-9*
applications
  checking package dependencies, *6-8*
  compatibility, *7-3*
  linked and upgrading, *7-6*
  linking with newer libraries, *7-7*
  running against older server, *7-6*
  upgrading, *7-1*
    client/server configurations, *7-3*
    compatibility rules, *7-4*
    options, *7-8*
    relinking rules, *7-4*
apxrelod.sql file
  reloading after downgrade, *8-5*
ArchiveLogTarget deprecated, *10-6*
ARGUMENTS user views
  changes to, *10-20*
ASM_PREFERRED_READ_FAILURE_GROUPS, *6-39*
asmcmd pwcreate deprecated, *10-8*
attributes
  xdb defaultTableSchema (deprecated), *10-53*
  xdb maintainOrder (deprecated), *10-53*
  xdb mapUnboundedStringToLob (deprecated), *10-53*
  xdb maxOccurs (deprecated), *10-53*
  xdb SQLCollSchema (deprecated), *10-53*
  xdb SQLSchema (deprecated), *10-53*
  xdb srclang (deprecated), *10-53*
  xdb storeVarrayAsTable (deprecated), *10-53*
  xdb translate (deprecated), *10-53*
auditing, *6-28*
  about transferring audit records after upgrade, *6-34*
  transferring unified audit records after upgrade, *6-35*
  unified auditing migration
    about, *6-28*
    audit options, *6-29*
    documentation references for non-unified auditing, *6-33*
    managing earlier audit records after migration, *6-32*
    procedure, *6-30*
    removing unified auditing, *6-32*
Automatic Diagnostic Repository (ADR), *1-21*
automatic undo management
  migrating to, *6-27*
Autoupgrade
  and error `Bad version number in .class file`, *3-3*
  Java 8 and later Java releases, *3-3*
  jobid, *3-6*
AutoUpgrade, *3-2*
autoupgrade -config_values-
  -console, *3-21*
  -debug, *3-21*
  -mode, *3-21*
  -noconsole, *3-21*
  -restore_on_fail, *3-21*
  -zip, *3-21*
  clear_recovery_data, *3-21*
autoupgrade -config-
  -console, *3-20*

autoupgrade -config- *(continued)*
-debug, *3-20*
-mode, *3-20*
-noconsole, *3-20*
-restore_on_fail, *3-20*
-zip, *3-20*
clear_recovery_data, *3-20*
autoupgrade -preupgrade, *3-31*
AutoupgradeOracle Database 11g Releease 2 and earlier releases, *3-3*

B

backing up the database, *2-44*
backup, *2-44*
backups
after upgrading, *6-19*
before downgrading, *8-4*
Zero Data Loss Recovery Appliance restriction, *10-45*
benchmarks, *2-12*
benefits of options for upgrading precompiler and Oracle Call Interface (OCI) applications, *7-8*
BFILE
migrating to, *6-28*
BLOB
migrating to, *6-28*
BP (bundle patches, *2-19*, *6-1*
broker configuration
exporting, *2-15*

C

capturing and replaying database workload, *2-9*
case sensitivity
for passwords, *6-22*
catcon
and downgrades, *8-27*
running SQL commands with, *8-29*
catcon.pl, *4-24*, *4-79*, *8-10*, *A-1*
using to check PDBs, *A-11*
catctl.pl, *4-15*, *4-17*, *A-1*
running shell commands to start, *4-14*
catdwgrd.sql, *A-1*
catdwgrd.sql script, *8-10*
catnoamd.sql, *A-1*
catnoexf.sql, *A-1*
catrelod.sql, *8-18*
CATRELOD.SQL script, *8-10*, *8-18*
catupgrd.log files
for CDBs and PDBs, *4-11*
catupgrd.sql
replaced by catctl.pl, *4-72*
CATUPGRD.SQL script
desupported.
See CATCTL.PL script manual upgrade and
catuppst.sql, *A-1*
CDB_JAVA_POLICY, *5-7*
CDBs, *4-79*, *B-1*
catctl.pl log files, *4-11*
downgrading, *8-5*
Oracle Label Security, *4-79*
rerunning the upgrade for CDB and PDBs, *4-6*
rerunning upgrades, *4-5*
restarting from a failed phase, *4-11*
restarting upgrades, *4-12*
upgrade scenarios, *4-22*
upgrading, *4-23*
using catcon with, *8-29*
change passwords
for Oracle-supplied accounts, *6-41*
changing PDB upgrade priority, *4-30*
changing scripts to use new features, *7-11*
client and server
configurations and upgrading, *7-3*
client software
upgrading, *7-6*
client-server configurations, *1-21*
client-side dynamic library, *7-7*
CLOB
migrating to, *6-28*
clone.pl
deprecated, *10-8*
CLUSTER_DATABASE initialization parameter, *2-30*, *5-2*
command-line upgrade
See manual upgrade
commands
orabase, *8-7*
orabasehome, *8-7*
oracle_home, *8-7*
compatibility
applications, *7-3*
between Oracle releases, *1-10*
checking for incompatibilities, *8-4*
COMPATIBLE initialization parameter, *1-14*
downgrading, *1-16*
overview for Oracle Database, *1-13*
COMPATIBLE initialization parameter, *1-14*, *6-39*
and PDB compatibility, *1-14*
checking the level of, *1-17*
considerations for downgrading, *1-16*
default, *1-14*
initial Java delay, *1-14*
Oracle recommendation, *1-14*
setting, *6-44*
values, *1-16*

component status, *5-9*
compression
  sqlnet.ora file parameters and, *2-20*
compression scheme, *2-20*
  SQLNET.COMPRESSION, *2-20*
  SQLNET.COMPRESSION_LEVELS, *2-20*
  SQLNET.COMPRESSION_THRESHOLD, *2-20*
configuration files
  copying, *2-30*
continuous_mine option desupported, *10-14*
copying configuration files, *2-30*
CREATE pfile FROM spfile, *2-30*
CREATE TABLE AS, *1-2*
crsuser deprecated, *10-33*
cursor cache, SMB, *2-10*

D

data copying
  using Export/Import, *9-1*
data definition language (DDL), *1-18*
data dictionary
  about changes that affect applications, *7-3*
  checking the state of, *6-2*
Data Pump Export/Import
  advantages of using, *2-5*
  recommendations, *9-1*
  versus Original Export/Import, *9-1*
  when to use, *9-1*
  with subsets of production database, *2-38*
Database Links, *A-23*
Database Replay
  database workloads before upgrading, *2-9*
Database Upgrade Assistant (DBUA)
  `-executePrereqs` option, *10-37*
  advantages, *2-4*
  and multitenant architecture upgrades, *10-37*
  CDBs and, *B-1*
  command-line options for, *B-19*
  guaranteed restore points, *B-1*
  noninteractive (silent) command-line syntax, *B-19*
  Pause and Continue, *10-37*
  PDBs and, *B-1*
  registering the database in the listener.ora file, *2-20*
  removed features, *10-37*
  running, *B-1*
  silent mode, *B-19*
  standalone prerequisite checks, *10-37*
  starting, *B-3*, *B-6*
Database Upgrade Assistant (DBUA) new features, *10-37*
Database XE, *1-27*
databases
  downgrading, *8-7*
  downgrading and Oracle Internet Directory registration, *B-1*
  downgrading manually, *8-10*
  upgrading the client software, *7-6*
DataGuardSyncLatency deprecated, *10-6*
datatypes, *10-66*
DBA_ACL_NAME_MAP, *10-39*
DBA_REGISTERED_MVIEW_GROUPS
  desupported, *10-59*
DBA_REGISTRY view, *5-9*
dbdowngrade utility, *8-7*
DbFileNameConvert deprecated, *10-6*
DBMS_DEBUG
  deprecated, *10-48*
DBMS_DST package, *A-24*
DBMS_LDAP package, *A-22*
DBMS_LDAP.init parameter, *A-22*
DBMS_NETWORK_ACL_ADMIN
  deprecated, *10-39*
DBMS_NETWORK_ACL_ADMIN package, *A-23*
DBMS_NETWORK_ACL_ADMIN_APPEND_HOST_ACE, *10-39*
DBMS_PDB package, *4-75*
DBMS_ROLLING, *10-38*
DBMS_STATS package
  upgrading statistics tables, *6-9*
DBMS_STATS.GATHER_DICTIONARY_STATS, *8-27*
DBMS_STATS.GATHER_DICTIONARY_STATS procedure, *8-29*
DBMS_STATS.GATHER_FIXED_OBJECTS_STATS, *8-28*
DBMS_XMLQUERY, deprecated, *10-27*
DBT error messages, *10-37*
DBUA
  See Database Upgrade Assistant
dbupgdiag.sql, *6-2*
dbupgrade, *4-14*, *4-15*, *A-1*
  manual upgrade and, *4-79*
dbupgrade shell command
  arguments for, *4-17*
dbupgrade.cmd, *4-15*, *A-1*
defaultTableSchema attribute (deprecated), *10-53*
demo
  replace in read-only Oracle homes, *6-14*
deprecated features in 18c, *10-25*
deprecated initialization parameters, *10-45*
deprecated parameters
  identifying, *10-63*, *10-80*
deprecated parameters and desupported parameters display, *2-30*
DESCRIBE procedure, *4-75*

desupported initialization parameters, *10-55*
desupported parameters
  identifying, *10-63*, *10-80*
developer applications
  upgrading forms and, *7-11*
DGMGRL, *10-38*
diagnostic data, *1-21*
direct upgrade, *1-3*
disk group compatibility, *6-39*
disks
  specifying preferred read failure groups, *6-39*
downgrades
  Oracle Data Guard broker configuration file
    backup, *2-15*
  reapplying patches after, *8-21*
downgrading
  and gathering dictionary statistics, *8-27*
  backing up your database, *8-4*
  CATRELOD.SQL, *8-10*, *8-18*
  checking for incompatibilities, *8-4*
  database links, *A-23*
  location of downgrade log files, *8-7*
  Oracle Enterprise Manager and, *8-22*
  ORADIM and, *8-10*
  patchset releases, *8-2*
  regathering fixed object statistics, *8-28*
  regathering stale statistics, *8-29*
  scripts, *8-7*, *8-10*
    rerunning, *8-10*
DV_PUBLIC
  deprecated, *10-9*
DV_REALM_OWNER
  deprecated, *10-9*
DV_REALM_RESOURCE
  deprecated, *10-9*
dvsys.dbms_macadm.enable_dv(), *6-15*

E

em_present, *4-50*
emca -restore command, *8-22*
emdwgrd, *A-1*
emdwgrd utility, *8-22*
emremove.sql, *A-1*
emulation, *2-39*
enforcing case-sensitivity for passwords, *6-22*
environment variables
  required for upgrading, *4-24*, *4-79*
exafusion_enabled desupported, *10-16*
exclusion lists
  about, *4-31*
  and PDB upgrades, *4-49*
  resume, *4-6*
Export utility, *9-1*
  data copying, *9-1*
export/import
  advantages and disadvantages, *2-5*
  benefits, *2-5*
  effects on upgraded databases, *2-5*
  recommendations, *9-1*
  time requirements, *2-6*
exporting
  broker configuration, *2-15*
extended datatype support
  deprecated, *10-26*
extended distance cluster configurations
  preferred read disks, *6-39*
extents
  reading from secondary, *6-39*
externally authenticated SSL users, *6-10*
extusrupgrade script, *6-10*

F

failed phases, *4-2*
Fast Recovery Area, *5-4*
fine-grained access control
  network utility packages, *A-22*
Flex Cluster architecture deprecated, *10-29*
Forms
  upgrading Oracle Forms applications, *7-11*
Full Transportable Export/Import, *4-73*

G

GET_MODEL views
  deprecated, *10-26*

H

Hardware Assisted Resilient Data (HARD)
  upgrading systems, *6-44*
HttpUriType type, *A-22*

I

IFILE (include file), *2-30*
  editing entry, *2-30*
    See also text initialization parameter file (PFILE)
image
  install, *1-29*, *7-7*, *10-19*
Import utility, *9-1*
  data copying, *9-1*
inclusion list, *4-10*
inclusion lists
  about, *4-31*
incompatibilities
  checking for, *8-4*

init.ora
and SGA permissions, *10-38*
initialization parameters, *2-30*
adjusting, *2-30*, *6-43*
ASM_PREFERRED_READ_FAILURE_GROUPS, *6-39*
COMPATIBLE, *1-14*, *6-39*
See also server parameter file (SPFILE)
install logs
component upgrade script, *5-9*
installation
Oracle Database software, *2-16*, *2-17*
instances
starting after a downgrade, *8-10*
Intelligent Data Placement (IDC)
deprecated, *10-48*
interim upgrade, *1-3*
intermediate releases
interim upgrading, *1-3*
interoperability, *1-17*
invalid objects
and utlrp.sql, *1-18*
recompiling, *6-5*
utlrp.sql script and, *4-24*, *4-79*, *5-9*, *8-10*
INVALID objects, *5-3*, *5-4*
Invalid objects after downgrades, *8-18*
INVALID status
component status, *5-9*

K

knowledge base, *1-5*

L

listener.ora file, *2-20*
modifying, *2-20*
listeners, *6-45*
and Oracle RAC upgrades, *6-37*
modifying with Oracle Net Configuration Assistant, *2-20*
load
level of concurrent demand when upgrading, *2-12*
load testing, *2-12*
log_archive_dest max_connections attribute deprecated, *10-16*
LOG_ARCHIVE_LOCAL_FIRST desupported, *10-55*
LogArchiveFormat deprecated, *10-6*
LogArchiveMaxProcesses deprecated, *10-6*
LogArchiveMinSucceedDest deprecated, *10-6*
LogArchiveTrace deprecated, *10-6*
LogFileNameConvert deprecated, *10-6*
logical standby databases
rolling upgrades, *1-23*
LogMiner
CONTINUOUS_MINE deprecated, *10-49*
lsnrctl command
Oracle Grid Infrastructure home and, *2-20*

M

maintainOrder attribute (deprecated), *10-53*
manual fixups
examples, *4-50*
manual upgrade, *1-2*, *2-4*, *6-45*
advantages, *2-4*
backing up the database, *2-44*
OCR configuration, *6-43*
manual upgrade example, *4-49*
manual upgrades
rerunning or restarting, *4-2*
mapUnboundedStringToLob attribute (deprecated), *10-53*
matches Oracle XQuery function (deprecated), *10-53*
matches XQuery function, *10-53*
max_connections desupported, *10-14*
MAX_SHARED_TEMP_SIZE
desupported, *10-60*
maxOccurs attribute in xdb namespace (deprecated), *10-53*
migrating
defined, *1-1*
migrating data, *1-1*
to a different operating system, *1-28*
migrating listener from Oracle home with lsnrctl command, *2-20*
moving data with Data Pump Export/Import, *9-1*
Mulitenant
restarting upgrades, *4-12*
Multiple Oracle Homes Support
advantages, *1-21*
multitenant
transferring unified audit records after upgrade, *6-35*
multitenant architecture
parallel upgrade of, *4-39*
multitenant architecture databases
upgrade scenarios, *4-22*
Multitenant architecture databases
and PDB COMPATIBILITY parameter setting, *1-14*
multitenant container database, *4-24*
multitenant container databases
See CDBs
multitenant databases
setting upgrade priorities with lists, *4-31*

multiversioning, *1-21*
mv_refresh, *4-50*
My Oracle Support, *1-5*
  knowledge base, *1-5*

N

NCLOB
  migrating to, *6-28*
network names and listeners, *6-45*
networks
  granting ACL access to network utility packages, *A-22*
new features
  adding after upgrade, *6-26*
  changing scripts to use and, *7-11*
new features, learning about, *2-3*
NO AUTHENTICATION status accounts, *2-13*, *7-2*
NO SCRIPT status, *5-9*
non-CDB architecture
  deprecated, *10-49*
non-CDB to CDB and PDB upgrades, *1-23*
non-CDBs, *B-1*
noncdb_to_pdb.sql script, *4-76*
not relinking upgraded application, *7-9*
NVARCHAR2 datatype
  EXTENDED, *10-66*

O

OCI applications
  changing, *7-11*
  changing to use new features, *7-11*
  dynamically-linked, *7-7*
  statically-linked, *7-7*
  upgrade and linking, *7-7*
  upgrading, *7-8*
  upgrading options, *7-8*
OCR, *10-58*
OFA, *1-19*, *1-21*
  See also Optimal Flexible Architecture
olspreupgrade.sql, *A-1*
operating system
  migrating data to, *1-28*
operating system requirements, *1-28*
Optimal Flexible Architecture, *1-19*, *1-21*
  about, *1-21*
    See also OFA
optimizer statistics
  regathering after downgrade, *8-29*
OPTIMIZER_ADAPTIVE_PLANS, *10-36*
OPTIMIZER_ADAPTIVE_STATISTICS, *10-36*
OPTION OFF status, *5-9*
options for upgrading precompiler and Oracle Call Interface (OCI) applications, *7-8*
ORA_TZFILE
  unsetting after downgrade, *8-5*
ORA-00336 log file size xxxx blocks error, *5-2*
ORA-00401 value for parameter compatible error, *5-2*
ORA-00600 Internal Error, *8-5*
ORA-00600: Internal Error Code
  and Oracle Data Pump, *8-34*
ORA-00704: bootstrap process failure, *5-2*
ORA-00904 "TZ_VERSION" invalid identifier error, *5-4*
ORA-00942 table or view does not exist error, *5-4*
ORA-01092: ORACLE instance terminated. Disconnection forced, *5-2*
ORA-01562 failed to extend rollback segment number error, *5-4*
ORA-01650: unable to extend rollback segment, *5-4*
ORA-01651: unable to extend save undo segment, *5-4*
ORA-01652: unable to extend temp segment, *5-4*
ORA-01653: unable to extend table, *5-4*
ORA-01654: unable to extend index, *5-4*
ORA-01655: unable to extend cluster, *5-4*
ORA-01722 invalid number error, *5-4*
ORA-03134: Connections to this server version are no longer supported., *10-42*
ORA-04031 unable to allocate nnn bytes of shared memory error, *5-4*
ORA-1017 invalid username/password, *10-42*
ORA-17500: ODM err:Operation not permitted, *10-67*
ORA-19815 WARNING
  db_recovery_file_dest_size error, *5-4*
ORA-20001: Downgrade cannot proceed, *8-30*
ORA-24247
  network access denied by access control list (ACL) error, *A-22*
ORA-24247: network access denied by access control list (ACL), *6-15*, *A-22*
ORA-27248: sys.dra_reevaluate_open_failures is running, *5-14*
ORA-28040 "No matching authentication protocol, *2-21*, *2-22*
ORA-28040 No matching authentication protocol., *10-42*
ORA-28040: No matching authentication protocol, *6-17*
ORA-28365, *5-7*
ORA-28365: wallet is not open, *2-19*
ORA-29283: Invalid File Option, *10-23*

ORA-32004: obsolete or deprecated parameters, *10-63*, *10-80*
ORA-39700: database must be opened with UPGRADE option, *5-2*
ORA-39701 database must be mounted EXCLUSIVE error, *5-2*
ORA-39709: Incomplete component downgrade, *8-10*, *8-18*
ORA-39709: incomplete component downgrade; string downgrade aborted, *8-30*
Oracle ACFS Encryption
  deprecated on Oracle Solaris and Microsoft Windows, *10-10*
Oracle ACFS Replication protocol REPV1
  deprecated, *10-10*
Oracle Administration Assistant tool, *10-30*
Oracle Application Express
  apexrelod.sql file, *8-5*
  update, *6-13*
Oracle Application Express configuration, *6-13*
Oracle ASM
  change in Oracle home location, *6-26*
Oracle Automatic Storage Management
  disk group compatibility, *6-39*
  password file (PWFILE), *5-12*
  preferred read failure groups, *6-39*
  rolling upgrades and, *1-23*
Oracle base, *1-21*
Oracle Call Interface (OCI
  upgrading applications and, *2-41*
Oracle Change Data Capture
  See Oracle GoldenGate
Oracle Cluster Registry (OCR)
  upgrading manually, *6-43*
Oracle Data Guard
  and AutoUpgrade, *3-63*
  change in properties storage, *10-2*
  rolling upgrades, *1-23*
  rolling upgrades with DBUA, *2-29*
Oracle Data Mining
  Model Details views, *10-26*
Oracle Data Provider for .NET
  desupport of Oracle.ManagedDataAccessDTC.dll, *10-32*
Oracle Data Pump, *1-2*
Oracle Database clients
  backup restrictions, *10-45*
Oracle Database Enterprise Edition
  converting from Enterprise Edition to Standard Edition, *1-27*
Oracle Database Express Edition, *1-27*
  recommended tasks after upgrade, *6-40*
  upgrading to Oracle Database, *1-27*
Oracle Database Standard Edition
  converting to Enterprise Edition, *1-25*
Oracle Database Vault, *2-16*, *2-17*, *8-33*
  enable after upgrade, *6-15*
  upgrading, *6-15*
Oracle Database XE, *1-27*
  upgrading to Oracle Database, *1-27*
Oracle Enterprise Manager Cloud Control
  upgrading with, *1-25*
Oracle Fleet Patching and Provisioning, *10-43*
Oracle FPP, *10-43*
Oracle GoldenGate, *10-75*, *10-87*
Oracle Grid Infrastructure
  file locations for OCR and voting disks, *10-58*
Oracle home
  copying configuration files from, *2-30*
  multiple, *1-20*
  ORACLE_HOME database directory on Windows, *2-30*
  ORACLE_HOME dbs directory on Linux or UNIX, *2-30*
  out-of-place requirement, *2-6*
Oracle Home User, *4-57*
Oracle Label Security, *2-16*, *2-17*
Oracle Layered File System, *10-43*
Oracle Multimedia Java APIs
  deprecated, *10-51*
Oracle Multitenant
  upgrade errors, *5-13*
Oracle Multitenant upgrades, *4-23*, *4-45*
Oracle Names support, *10-89*
Oracle Net Configuration Assistant, *2-20*
Oracle Optimal Flexible Architecture
  See Optimal Flexible Architecture
Oracle Optimizer
  and DBMS_STATS, *6-22*
Oracle RAC
  desupported on Oracle Database Standard Edition 2 (SE2), *10-15*
Oracle Real Application Clusters, *1-18*
Oracle release numbers, *1-11*
Oracle RMAN, *2-44*
  backing up the database, *2-44*
  running, *2-44*
Oracle Streams
  terminal release of, *10-34*
Oracle Text
  MAIL_FILTER, *10-28*
  Upgrading, *6-42*
  widened token columns, *6-33*
Oracle Text-supplied knowledge bases
  upgrading and, *6-12*
Oracle Universal Installer, *1-2*, *2-16*, *2-17*
Oracle update batching size disabled, *10-41*

Oracle wallet
preupgrade step for, 2-19
upgrading, 5-7
Oracle-supplied accounts
change passwords, 6-41
oracle.dbc.OracleConnection
deprecated, 10-50
oracle.jdbc.rowset
deprecated, 10-50
ORADIM, 4-57
downgrading and, 8-10
upgrading and, 4-24, 4-79
orapwSID password file, 2-30
Original Export/Import
versus Data Pump Export/Import, 9-1
OUI
See Oracle Universal Installer

P

Parallel Upgrade Utility, 4-15, 5-2
about, 4-72
and ability to upgrade schema-based tablespaces, 2-26
manual upgrade and, 4-79
rerunning upgrades, 4-2
restarting, 4-2
resume option, 4-2
running on specified CDBs, 4-6
See Also catctl.pl, dbupgrade, dbupgrade.cmd, 4-72
setting tablespaces to READ ONLY, 10-41
PARALLEL_ADAPTIVE_MULTI_USER deprecated, 10-45
PARALLEL_AUTOMATIC_TUNING desupported, 10-55
PARALLEL_IO_CAP_ENABLED desupported, 10-55
PARALLEL_SERVER desupported, 10-55
PARALLEL_SERVER_INSTANCE desupported, 10-55
parameter file
and permissions to read and write the SGA, 10-38
backing up, 2-30
password verifiers, 2-21
password versions, 2-22
passwords
10G password version, finding and resetting, 6-23
case sensitive, 6-22
patch set updates, 2-19, 6-1
patchset releases
downgrading, 8-2
Pause and Continue, 10-37
PDB
COMPATIBILITY parameter and CDB, 1-14
PDB upgrades after CDB upgrade, 4-49
PDB$SEED
counted as one PDB during upgrades, 4-20
PDBs, 4-79, B-1
catupgrd.log files, 4-11
downgrading, 8-5
moving non-CDBs into, 4-75
Oracle Label Security, 4-79
pluggable upgrades of, 10-37
priority-based PDB upgrades, 10-37
rerunning upgrades, 4-5
restarting from a failed phase, 4-11
restarting upgrades, 4-12
setting upgrade priorities with lists, 4-31
upgrade errors, 5-13
upgrade scenarios, 4-22
upgrades of, 10-37
upgrading, 4-23
upgrading individually, 4-45
upgrading using priority lists, 4-30
performance
unified audit trail, 6-34
PFILE
See text initialization parameter file (PFILE)
physical standby database
rolling upgrades, 1-23
PL/SQL packages
checking, 6-8
placement on shared storage deprecated, 10-58
pluggable databases
See PDBs
Pluggable Databases
unified auditing migration and, 6-30
Post-Upgrade Status Tool, 6-2
rerunning upgrades, 4-2
postupgrade status tool
warning, 5-4
postupgrade_fixups.sql, A-1, A-7, A-11
depend_usr_tables, 4-65
example, 4-65
old_time_zones_exist, 4-65
post_dictionary, 4-65
Pre-Upgrade Information Tool, A-5
and postupgrade_fixups.sql, 6-19
command syntax for, A-8
deprecated parameters and desupported parameters display, 2-30
example on Windows, 4-50
warnings and actions to take, A-21
working with, A-21
See also preupgrade.jar
precompiler application
changing to use new features, 7-10

precompiler applications
upgrading and, 2-41
precompilers
applications
changing, 7-10
upgrading options, 7-8
upgrading applications, 7-8
preferred read failure groups
setting up, 6-39
preprocessing script
OLS and DV, A-1
preupgrade directory, A-5
preupgrade steps, 2-4
preupgrade_fixups.sql, 4-50, A-6, A-11
about, A-5
postupgrade_fixups.sql
about, A-5
preupgrade.jar, 4-15, 4-24, A-1, A-5, A-13, B-3
and PDBs, A-8
command syntax for, A-8
setting user environment variables for, A-8
See also Pre-Upgrade Information Tool
preupgrade.log, 4-50, A-11
preupgrd.jar, 5-3
preupgrd.sql, 5-3
obsolete in 12.2, A-1
priority lists, 4-30
about, 4-31
test upgrades using, 2-39
PRKH-1014 error, 2-16, 2-17
Pro*C/C++ applications, 2-8
PRODUCT_USER_PROFILE
deprecated, 10-29
proxy PDBs
upgrades, 4-23
PSU, 2-19, 6-1

R

Rapid Home Provisioning, 4-22
upgrades using, 1-2
RAW datatype
EXTENDED, 10-66
raw devices
desupported, 10-66, 10-86
OCFS
desupported on Windows, 10-66
RDBMS DST patch, A-24
read-only Oracle homes
replace demo directory, 6-14
read-only tablespaces, 2-26
recompiling invalid objects, 4-24, 4-79, 6-5, 8-10
on a CDB, 6-6
recovery catalog
upgrading, 6-9
release numbers, 1-11
Release Update (Update, 2-19, 6-1
release update (Update), 1-11
release update revision (Revision), 1-11
Release Update Revision (Revision), 2-19, 6-1
releases
definition, 1-11
multiple, 1-21
upgrade paths, 1-3
REMOVED status, 5-9
replace Oracle XQuery function (deprecated), 10-53
replace XQuery function, 10-53
rerunning upgrades
multitenant database architecture, 4-5
reserved words
additions and applications, 7-3
resuming upgrades, 4-2
RMAN
See Oracle RMAN
rollback segments
migrating to automatic undo management, 6-27
Rolling Upgrade Using Active Data Guard, 1-25
rolling upgrades
Oracle Clusterware and, 1-23
physical standby database, 1-23
rolling upgrades
with SQL Apply and logical standby databases, 1-23
SQL Apply
rolling upgrades, 1-23
summary of methods, 1-23
rootupgrade.sh script, 2-16, 2-17
rpath option for linking, 7-7
rpm –ivh, 10-19
RPM-based database installation, 10-19
run-time library search path, 7-7
running multiple Oracle releases, 1-21

S

schema-only accounts, 2-13, 7-2
scripts
checking the Oracle Data Dictionary state, 6-2
downgrading, 8-7, 8-10
manual upgrade and, 4-24, 4-79
SEC_CASE_INSENSITIVE_LOGON, 10-42
security
case-sensitive passwords, 6-22
server
compatibility rules, 7-5
server parameter file (SPFILE), 2-30
migrating to, 6-42

server parameter file (SPFILE) *(continued)*
upgrading systems with HARD-compliant storage, *6-44*
services
migrating, *10-19*
Single Client Access Names (SCAN), *6-37*
SP2-0152: ORACLE may not be functioning properly, *8-5*
SP2-1503: Unable to initialize Oracle call interface, *8-5*
SP2-1540 "Oracle Database cannot startup in an Edition session" error, *5-6*
SPFILE
See server parameter file (SPFILE)
spool upgrade results to a log file, *A-5*
SQL execution plans, *2-12*
SQL Management Base (SMB), *2-10*
cursor cache, *2-10*
SQL Performance Analyzer, *2-10*
SQL plan baselines
unpacking, *2-12*
SQL plan management, *2-10*
SQL Management Base and, *2-10*
SQL queries
testing, *2-12*
SQL workload, *2-10*
SQL_92_SECURITY default change, *10-36*
SQL*Plus
product-level security deprecated, *10-29*
scripts
upgrading, *7-11*
SQL/MM still image standard support, *10-51*
SQLCollSchema attribute (deprecated), *10-53*
SQLJ
client-side support only, *10-59*
deprecation in server, *10-76*
SQLNET.ALLOWED_LOGON_VERSION_SERVER, *2-21*, *2-22*
sqlnet.ora file
compression and, *2-20*
SQLSchema attribute (deprecated), *10-53*
srclang attribute (deprecated), *10-53*
SSL external users conversion, *6-10*
staging table
creating, *2-12*
Standard Edition
Export utility, *1-27*
starter database, *5-12*
standard edition high availability
guidelines, *2-28*
standard operating environment, *10-43*
StandbyFileManagement deprecated, *10-6*
STARTUP UPGRADE command, *8-10*
statically linked Oracle client-side library code, *7-7*
statistics tables
upgrading, *6-9*
status
NO SCRIPT, *5-9*
OPTION OFF, *5-9*
REMOVED, *5-9*
UPGRADED, *5-9*
status INVALID
component status, *5-9*
storeVarrayAsTable attribute (deprecated), *10-53*
support
See My Oracle Support
support note 472937.1, Information On Installed Database Components, *A-21*
support note 753041.1, How to Diagnose Components with NON VALID Status, *A-21*
support note ID 412160.1, for RDBMS DST patches, *A-24*
symbolic link
Oracle Universal Installer and, *7-7*
syntax check
application code, *7-9*
SYS.LINK$ table, *A-23*
system global area
permissions to read and write, *10-38*

T

tablespaces
read-only, *2-26*
testing
applications for upgrade, *2-13*, *6-36*
developing a plan, *2-6*
functional for upgrade, *2-8*
high availability for upgrade, *2-8*
integration for upgrade, *2-8*
minimal for upgrade, *2-7*
multitenant architecture upgrades, *2-39*
performance for upgrade, *2-9*
upgraded test databases, *2-13*
using Database Replay, *2-9*
using priority list emulation, *2-39*
volume/load stress for upgrade, *2-12*
testing the upgrade process, *2-38*
text initialization parameter file (PFILE), *2-30*
time zone file
how to resolve a mismatch, *A-24*
unsetting after downgrade, *8-5*
TIMESTAMP WITH TIME ZONE data type, *A-24*
TIMESTAMP WITH TIMEZONE data type, *A-24*
tnsnames.ora
adjusting after manual upgrades, *6-45*
token limitations, *10-20*
training, where to find, *2-3*

translate attribute (deprecated), *10-53*
Transparent Data Encryption
and upgrading wallets, *2-19*
Transparent Data Encryption (TDE)
and Oracle wallet upgrades, *5-7*
Transportable Export/Import, *4-73*
trgowner_no_admndbtrg, *4-50*
troubleshooting
and using resume, *4-2*
authentication protocol errors, *2-21*, *2-22*
AutoUpgrade, *3-8*
Autoupgrade utility, *3-73*
bringing up tablespaces after catastrophic upgrade failures, *2-26*
catupgrd.sql deprecation error, *10-62*
CDB_JAVA_POLICY errors, *5-7*
datapatch failures, *5-14*
EDITION session error, *5-6*
Flash Recovery Area, *5-4*
INVALID objects, *5-3*, *5-4*
ORA-00600 during Oracle Data Pump exports, *8-34*
ORA-00600 error with database link passwords, *8-34*
ORA-00942 table or view does not exist, *5-4*
ORA-03134: Connections to this server version are no longer supported., *10-42*
ORA-1017 invalid username/password, *10-42*
ORA-20000, *8-33*
ORA-31011: XML parsing failed, *8-33*
ORA-39709, *8-18*
ORA-45415, *2-29*
ORA-65394 runtime error, *10-20*
Oracle Database Vault downgrades, *8-33*
Oracle Internet Directory
and downgrades to earlier releases, *B-1*
PDB upgrades, *4-49*
PLS-1919 compile time error, *10-20*
REMOTE_LOGIN_PASSWORDFILE warning, *6-41*
restore scripts and Oracle Internet Directory registration, *B-1*
rollback segments/undo tablespace, *5-4*
running out of resources, *5-4*
services running in old Oracle home after upgrade, *10-19*
shared memory, *5-4*
starting database in upgrade mode, *5-2*
SYSTEM and SYSAUX tablespaces, *5-4*
upgrade termination
due to ORA-00904, *5-4*
due to ORA-01722, *5-4*
upgrades, *5-1*
Troubleshooting
ORA-32004: obsolete or deprecated parameter, *10-63*, *10-80*
ORA-39709, *8-10*
troubleshooting the upgrade
termination due to ORA_00942, *5-4*
type of software upgrade, *7-3*

U

UNDO_MANAGEMENT initialization parameter, *6-27*
unicode collation algorithm 6.1
deprecated, *10-51*
unified audit trail
performance improvement, *6-34*
unified auditing, *6-28*
about transferring audit records after upgrade, *6-34*
transferring unified audit records after upgrade, *6-35*
See also auditing
upg_summary.rpt, *5-11*
upgrade
manual upgrade example, *4-49*
upgrade methods
choosing, *2-3*
Data Pump Export/Import, *2-5*
Database Upgrade Assistant, *1-2*
Database Upgrade Assistant (DBUA), *2-4*
emulation, *2-39*
manual, *2-4*
silent mode, *B-19*
upgrade path
determining, *1-3*
table, *1-3*
upgrade procedures
error messages, *10-37*
summary of major steps, *1-6*
upgrade process testing, *2-38*
and utlrp.sql, *1-18*
upgrade summary report
location of, *5-11*
upgrade.xml not found error, *5-3*
UPGRADED status, *5-9*
upgraded test databases, *2-13*
upgrading
applications, *7-1*
compatibility rules, *7-4*
options, *7-8*
relinking, *7-4*
defined, *1-1*
initialization parameters, *2-30*
new administrative procedures, *6-27*
Oracle Application Express, *6-13*

upgrading *(continued)*
Oracle Forms applications, *7-11*
ORADIM and, *4-24*, *4-79*
post upgrade actions, *6-1*
preparation, *2-2*
recovery catalog, *6-9*
scripts and manual upgrade, *4-24*, *4-79*
SQL*Plus scripts, *7-11*
statistics tables, *6-9*
testing, *2-6*
troubleshooting, *5-1*
using the Database Upgrade Assistant, *B-1*
where to find information about, *1-5*
upgrading a cluster database
setting the CLUSTER_DATABASE initialization parameter, *2-30*
UTL_FILE, *10-23*
UTL_FILE_DIR deprecated, *10-45*
UTL_INADDR package, *A-22*
UTL_MAIL package, *A-22*
UTL_SMTP package, *A-22*
UTL_TCP package, *A-22*
utlrp.sql, *1-18*, *6-5*, *A-1*
on a CDB, *6-6*
utlrp.sql script
for recompiling invalid objects, *4-24*, *4-79*, *8-10*
utluptabdata.sql, *10-41*
utlusts.sql, *5-4*, *5-9*, *6-2*, *A-1*

V

V$OPTION view, *5-9*
V$REPLPROP
desupported, *10-59*
V$REPLQUEUE
desupported, *10-59*
VARCHAR2 datatype
EXTENDED, *10-66*
VERIFY_FUNCTION
deprecated, *10-52*
VERIFY_FUNCTION_11G
deprecated, *10-52*
views
desupported, *10-59*
volume
amount of data upgraded, *2-12*
voting disk files
placement on shared storage deprecated, *10-58*

W

wallets
procedure to migrate, *2-19*
warning XDB now invalid error, *5-13*
Windows
and virtual accounts, *4-57*
manual upgrade, *4-50*, *4-57*, *4-65*
remote upgrades deprecated, *10-37*
using ORADIM with, *4-57*
workloads
capturing and replaying, *2-9*

X

xdb defaultTableSchema attribute (deprecated), *10-53*
xdb maintainOrder attribute (deprecated), *10-53*
xdb mapUnboundedStringToLob attribute (deprecated), *10-53*
xdb maxOccurs attribute (deprecated), *10-53*
xdb SQLCollSchema attribute (deprecated), *10-53*
xdb SQLSchema attribute (deprecated), *10-53*
xdb srclang attribute (deprecated), *10-53*
xdb storeVarrayAsTable attribute (deprecated), *10-53*
xdb translate attribute (deprecated), *10-53*
XE, *1-27*
XML DB
desupported functions and procedures, *10-59*
XQuery language
functions, *10-53*
matches, *10-53*
matches (deprecated, Oracle), *10-53*
replace (deprecated, Oracle), *10-53*

Z

Zero Data Loss Recovery Appliance
backup restriction, *10-45*
Zero Downtime upgrades, *4-22*