Oracle® Database

Database Upgrade Guide

21c
F17069-42
February 2025

Oracle Database Database Upgrade Guide, 21c

F17069-42

Copyright © 1996, 2025, Oracle and/or its affiliates.

Primary Author: Douglas Williams

Contributors: Frederick Alvarez, Mike Dietrich, Joseph Errede, Cindy Lim, Wei Lin , Byron Motta, Daniel Overby Hansen, Benjamin Speckhard, Carol Tagliaferri, Hector Vieyra, Philip Yam, Zhihai Zhang

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, software documentation, data (as defined in the Federal Acquisition Regulation), 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," "commercial computer software documentation," or "limited rights data" 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®, Java, MySQL, and NetSuite 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

2 Preparing to Upgrade Oracle Database

3 Upgrading Databases with Oracle Data Guard Standbys

4 Using AutoUpgrade for Oracle Database Upgrades

5 Upgrading Oracle Database Using Parallel Upgrade Utility or Replay Upgrade

6 Troubleshooting the Upgrade for Oracle Database

7 Postupgrade Tasks for Oracle Database

8 Upgrading Applications After Upgrading Oracle Database

9 Downgrading Oracle Database to an Earlier Release

10 Oracle Database Changes, Desupports, and Deprecations

A AutoUpgrade REST APIs

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

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* 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.

Convention	Meaning
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.

- Oracle Database Releases That Support Direct Upgrade
  Review the supported options for direct upgrades to Oracle Database 21c.
- 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 Database Releases
  To run multiple Oracle Database releases at the same time, follow Optimal Flexible Architecture (OFA) standards.
- 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
  Understand image-based installation to simplify installation and configuration of Oracle Database software.

Oracle Database Releases That Support Direct Upgrade

Review the supported options for direct upgrades to Oracle Database 21c.

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

- 19c
- 18c
- 12c Release 2 (12.2)

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 release 12.2, 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 21c.

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 11g Release 2 (11.2) then to upgrade to Oracle Database 21c, follow these steps:
>
> 1. Upgrade Release 11.2 to release 12.2, using the instructions in *Oracle Database Upgrade Guide 12c Release 2 (12.2)*, including running the preupgrade script for 12.2.
> 2. Upgrade Oracle Database 12c Release 2 (12.2) directly to Oracle Database 21c. Use the instructions in this book, *Oracle Database Upgrade Guide*, including running the preupgrade script for Oracle Database 21c.

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 21c.

Table 1-1 Examples of Upgrade Paths for Oracle Database 21c

Current Release	Upgrade Options
19 (all releases), 18 (all releases), 12.2.0.1	**Direct upgrade is supported.** Perform the upgrade using the current Oracle Database Upgrade Guide, which is this guide.
12.1.0.2. 12.1.0.1 11.2.0.1, 11.2.0.2, 11.2.0.3, 11.2.0.4 11.1.0.6, 11.1.0.7 10.2 or earlier releases	**Direct upgrade to Oracle Database 21c 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: • Releases 12.1.0.1, 12.1.0.2, 11.2.0.3, 11.2.0.4: Upgrade to Oracle Database 12c Release 2 (12.2), and then upgrade to Oracle Database 21c. • Releases 10.2.0.2, 10.2.0.3, 10.2.0.4, 10.2.0.5 or 10.1.0.5: Upgrade to release 11.2.0.3 or 12.1, and then to 12.2, and then to Oracle Database 21c. **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.

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.
- 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 the AutoUpgrade utility using the `preupgrade` parameter.

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:

- AutoUpgrade Utility

  Identifies issues before upgrades, deploys upgrades, performs postupgrade actions, and starts the upgraded Oracle Database.
- Manual upgrade using the Parallel Upgrade Utility, and other command-line utilities

  Enables upgrades to be performed using shell scripts.
- 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.

- Using Oracle Enterprise Manager Fleet Maintenance to upgrade databases.

  Fleet Maintenance, a component of Oracle Enterprise Manager, enables you to use the `emcli` command-line environment to automatically patch and upgrade a large number of databases in your enterprise. Database Fleet Maintenance enables administrators to maintain groups or pools of Oracle Homes and associated databases by applying database updates that include interim one-off patches, including quarterly security patch updates.

  Fleet Maintenance facilitates the creation of a gold image using a reference environment, or by updating existing gold images by adding desired patches (standard operating environments). With Fleet Maintenance, you can use those gold images, and deploy them across the enterprise for both patching and upgrade with a single `emcli` command. Fleet Maintenance comes with an extensive set of precheck and intelligent rollback options, and with a user interface that facilitates tracking deployment progress. If there is an issue with patching or upgrade databases, then the Fleet Maintenance user interface can display the specific logs for those databases.

  For more information about Fleet Maintenance, refer to *Oracle Enterprise Manager Cloud Control Database Lifecycle Management Administrator's Guide*

Related Topics

- Oracle Data Pump Import
- *Oracle Clusterware Administration and Deployment Guide*
- Database Fleet Maintenance

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 the AutoUpgrade utility using the `preupgrade` parameter.

Through its support website, My Oracle Support, Oracle provides late-breaking updates, discussions, and best practices about preupgrade requirements, upgrade processes, postupgrade tasks, compatibility, and interoperability.

Before you begin upgrades, Oracle strongly recommends that you download the latest version of the AutoUpgrade utility, which is available on My Oracle Support. The latest version contains the most recent checks and tests that Oracle can provide to assist you to prepare your system for upgrades, and to complete upgrades successfully. You can perform your upgrades directly using AutoUpgrade. If you do not use AutoUpgrade to perform the upgrade, then to identify potential upgrade issues, you must run AutoUpgrade using the `preupgrade` parameter in `analyze` mode. This AutoUpgrade option replaces the Pre-Upgrade Information Tool that you may have used in earlier releases.

Mike Dietrich's blog site, "Upgrade Your Database Now," Daniel Overby Hansen's site, "Databases Are Fun," can offer you the most current insights from the upgrade and migration product managers for Oracle. Their opinions and recommendations are based on their experiences and interactions with Oracle customers, and can save you time and effort. They are highly recommended.

We strive to provide you with the latest information available to us in this publication. If you have any suggestions for how we can improve, or comments on where we succeeded or failed, please feel free to comment directly on the **Was this page helpful?** links that you find at the bottom of each HTML page. Our mission at Oracle is to help you to succeed. The authors of this publication care what you think. If you think we can do something to make your work easier, and your upgrades successful, then please let us know.

Related Topics

- AutoUpgrade Tool (Doc ID 2485457.1)
- Oracle Database 19c Important Recommended One-off Patches (Doc ID 555.1)
- Upgrade Your Database Now
- Databases Are Fun

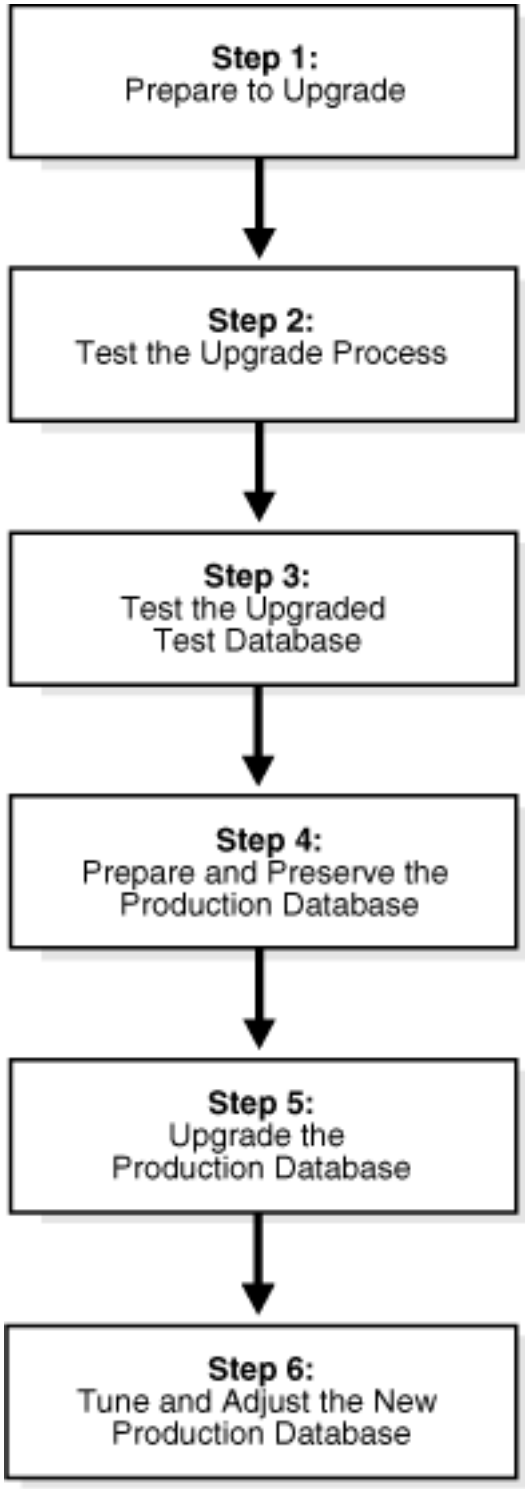
Major Steps in the Upgrade Process for Oracle Database

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 fixups, 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 Upgrade Topics
  Review to understand how statements in upgrade topics 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 Oracle Database Release Numbers

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

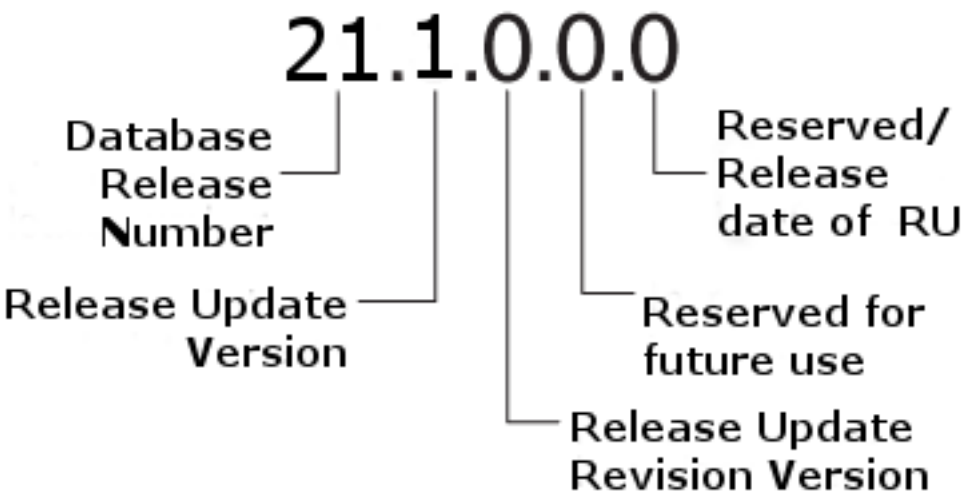
> **Note:**
>
> Oracle provides quarterly updates in the form of Release Updates (Updates, or RU) and Release Update Revisions (Revisions, or RUR). Oracle no longer releases patch sets or bundle patch sets. For more information, see My Oracle Support Note 2285040.1.

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, or RU).
- Third numeral: This numeral indicates the release update revision version (Revision, or RUR).
- Fourth numeral: This numeral is reserved for future use. Currently it is always set to 0.
- Fifth numeral: Although only the first three fields are commonly used, the fifth field can show a numerical value that redundantly clarifies the release date of a release update (RU), such as 19.7.0.0.200414.

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

> **Caution:**
>
> Oracle strongly recommends that you apply the most recent Release Update to your target databases before starting an upgrade, and before starting a downgrade. If possible, also ensure that your source environment is patched. Release updates are cumulative. If you are also updating an Oracle Grid Infrastructure environment, then always apply the latest Release Update to the Grid environment first before updating Oracle Database Oracle homes. For more information about updates, refer to My Oracle Support note 2118136.2.

Related Topics

- My Oracle Support note 2285040.1
- My Oracle Support note 2118136.2

Convention for Referring to Release Numbers in Upgrade Topics

Review to understand how statements in upgrade topics apply to releases.

When a statement is made in an Oracle Database upgrade topic about an annual Oracle Database release, that statement applies to the entire release. A statement about Oracle Database 21c, Oracle Database 19c, or any other release supported for direct upgrade, applies to all component-specific and platform-specific releases within that release, unless otherwise specified.

When a statement is made about a specific quarterly release update (RU, or Update), that statement applies to that quarterly Update.

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 updating 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.
- About the COMPATIBLE Initialization Parameter in Oracle Database
  Review to understand how to set the COMPATIBLE initialization parameter for non-CDB and multitenant architecture containers in Oracle Database 21c.
- Values for the COMPATIBLE Initialization Parameter in Oracle Database
  Review to find the default and minimum values for the COMPATIBLE initialization parameter for Oracle Database 21c.
- About Downgrading and Compatibility for Upgrading Oracle Database
  Before upgrading to Oracle Database 21c, you must set the COMPATIBLE initialization parameter to at least 12.2.0.
- 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 find the COMPATIBLE initialization parameter value set for your database.

Understanding Oracle Database Compatibility

If new features are incompatible with your earlier release, then updating 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 to the maximum value, you cannot subsequently downgrade the database to an earlier release.

If you can provide for an additional maintenance window after your upgrade window, then consider increasing the COMPATIBLE parameter 7 to 10 days after the upgrade, when you have run your applications on the upgraded system, and are sure that a downgrade is not required. If you do not have an additional maintenance window available, then either raise COMPATIBLE after the upgrade (and lose downgrade capability), or continue to use the earlier release

COMPATIBLE setting until you are near a maintenance window, so you have the option of downgrading available to you.

> **Note:**
>
> The value of the COMPATIBLE parameter should not be changed for a Release Update (RU) or Release Update Revision (RUR). For example, assume you are running Oracle Database 21c and the value of COMPATIBLE is 21.0.0. You then apply Oracle Database Release Update 21.4.0.0.0. Do not set the value of COMPATIBLE to 21.4.0; leave it set to 21.0.0.

About the COMPATIBLE Initialization Parameter in Oracle Database

Review to understand how to set the COMPATIBLE initialization parameter for non-CDB and multitenant architecture containers in Oracle Database 21c.

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

- Understanding the COMPATIBLE Initialization Parameter
  Learn about what the COMPATIBLE parameter does, when to raise the parameter release value, and what effects you can expect when you set the parameter to the new Oracle Database release.
- Rules for COMPATIBLE Parameter Settings in Multitenant Architecture
  When you plug in PDBs to a later release CDB, the CDB$ROOT COMPATIBLE parameter setting can change the PDB COMPATIBLE parameter, or prevent plug-ins.

Understanding the COMPATIBLE Initialization Parameter

Learn about what the COMPATIBLE parameter does, when to raise the parameter release value, and what effects you can expect when you set the parameter to the new Oracle Database release.

In Oracle Database 21c, when the COMPATIBLE initialization parameter is not set in your parameter file, the COMPATIBLE parameter value defaults to 21.0.0 If you do not set the COMPATIBLE initialization parameter to 21.0.0, then you cannot use the new Oracle Database 21c features, because your upgraded database is not running in the required COMPATIBILITY setting for Oracle Database 21c features.

Note the following restrictions for COMPATIBLE values:

- The minimum Oracle Database release supported for direct upgrade is Oracle Database 12c Release 2 (12.2).

  Before upgrading to Oracle Database 21c, you must set the COMPATIBLE initialization parameter to at least 12.2.0, which is the minimum compatible setting for Oracle Database 21c.
- In Oracle Real Application Clusters (Oracle RAC) environments, each Oracle Database instance in the Oracle RAC cluster must run with the same COMPATIBLE setting.
- The compatible parameter must be at least 3 decimal numbers, separated by periods. For example:

```
SQL> ALTER SYSTEM SET COMPATIBLE = '21.0.0' SCOPE=SPFILE;
```

Oracle strongly recommends that you only use three decimals in your COMPATIBLE setting.

- After you increase the COMPATIBLE parameter, you cannot downgrade the database, and you cannot flash back to restore points.

> **Caution:**
>
> Oracle recommends that you only raise the COMPATIBLE parameter to the current release level after you have thoroughly tested the upgraded database.
>
> After the COMPATIBLE parameter has been increased be aware that downgrade to your earlier release can be unsupported, even though upgrades from that release are supported.
>
> When you plug in an earlier release PDB to a later release CDB where COMPATIBLE is set to a later release than the earlier release PDB, and you upgrade the PDB by using an unplug/plug/upgrade procedure, the COMPATIBILE setting of the upgraded PDB is automatically updated to the COMPATIBLE setting of the later release CDB. The COMPATIBLE setting for PDBs must always be the same setting as CDB$ROOT.

Related Topics

- Managing Initialization Parameters Using a Server Parameter File

Rules for COMPATIBLE Parameter Settings in Multitenant Architecture

When you plug in PDBs to a later release CDB, the CDB$ROOT COMPATIBLE parameter setting can change the PDB COMPATIBLE parameter, or prevent plug-ins.

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 and minimum values for the COMPATIBLE initialization parameter for Oracle Database 21c.

Default and Minimum COMPATIBLE Parameter Values

The minimum supported release for direct upgrade to Oracle Database 21c is Oracle Database 12c Release 2 (12.2). The minimum COMPATIBLE parameter value for Oracle Database 21c is

12.2.0. The default value for the COMPATIBLE parameter is 21.0.0. Before you use a direct upgrade to Oracle Database 21c, you must set the COMPATIBLE parameter on your source Oracle Database release to at least 12.2.0.

The COMPATIBLE parameter should not be changed for a Release Update (RU) or a Release Update Revision (RUR), either for CDB or Non-CDB instances. The following table lists the default and minimum values for the COMPATIBLE parameter in Oracle Database 21c, compared to earlier releases supported for direct upgrade:

> **Caution:**
>
> After the COMPATIBLE parameter is increased, database downgrade is not possible.
>
> When you plug in an earlier release PDB to a later release CDB where COMPATIBLE is set to a later release than the earlier release PDB, and you upgrade the PDB by using an unplug/plug/upgrade procedure, the COMPATIBILE setting of the upgraded PDB is automatically increased to the COMPATIBLE setting of the later release CDB.
>
> Do not alter the COMPATIBLE parameter to a value other than a default release value. Use only one of the default values listed in the following table.

Table 1-2 The COMPATIBLE Initialization Parameter

Oracle Database Release	Default Value	Minimum Value
Oracle Database 21c	21.0.0	12.2.0
Oracle Database 19c	19.0.0	11.2.0
Oracle Database 18c	18.0.0	11.2.0
Oracle Database 12c Release 2 (12.2)	12.2.0	11.2.0

About Downgrading and Compatibility for Upgrading Oracle Database

Before upgrading to Oracle Database 21c, you must set the COMPATIBLE initialization parameter to at least 12.2.0.

After upgrading to Oracle Database 21c, 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 21c features are available while the COMPATIBLE initialization parameter is set to a lower value.

> **Caution:**
>
> After you increase the COMPATIBLE parameter to the current release, the database cannot be downgraded to an earlier release.

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 21c database with the `COMPATIBLE` initialization parameter set to `12.2.0`, then the database software generates database structures on disk that are compatible with Oracle Database Release 12c Release 2 (12.2). 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, some new features are enabled that do not create changes on disk that are incompatible with Oracle Database Release 12c Release 2.
- If you make changes to the database that make the database incompatible with the `COMPATIBLE` initialization parameter setting for the database, then does not start, and initialization terminates in an error. To resolve this issue, set the `COMPATIBLE` initialization parameter to a value that is equivalent to the setting required for the changes you made.

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

Checking the Compatibility Level of Oracle Database

Use this SQL query to find the `COMPATIBLE` initialization parameter value set for your database.

```
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 Running Multiple Oracle Database Releases

To run multiple Oracle Database releases at the same time, follow Optimal Flexible Architecture (OFA) standards.

- Organizing Oracle Software with Optimal Flexible Architecture
  Organize Oracle software binaries using the Optimal Flexible Architecture configuration guidelines.
- Interoperability of Oracle Database Client Releases with Oracle Database
  Review to understand which Oracle Database client versions are supported to work with different Oracle Database releases.
- 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.

Organizing Oracle Software with Optimal Flexible Architecture

Organize Oracle software binaries using the Optimal Flexible Architecture configuration guidelines.

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*.

Interoperability of Oracle Database Client Releases with Oracle Database

Review to understand which Oracle Database client versions are supported to work with different Oracle Database releases.

For information about client support with Oracle Database releases, see "Client / Server Interoperability Support Matrix for Different Oracle Versions (Doc ID 207303.1)"

Related Topics

- Client / Server Interoperability Support Matrix for Different Oracle Versions (Doc 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 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.
- Multiple databases, of different versions, owned by different users can coexist concurrently.
- To install Oracle Database software in multiple Oracle homes, you must extract the image file in each Oracle home, and then run the setup wizard from the respective Oracle home.

- 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 is the release number in the format of major and RU release number. For example, with the release number 21.3.0.0.0, the major release is 21 and the RU release number is 3.
- Later Oracle Database releases can access earlier Oracle Database releases. However, this access is only for upgrades. For example, Oracle Database 23c can access an Oracle Database 21c if the 21c database is started up in upgrade mode.
- 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.
- For information about release support timelines, refer to My Oracle Support Doc ID 742060.1

Related Topics

- My Oracle Support Note 742060.1

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.
- Overview of Steps for Upgrading Oracle Database Using Oracle GoldenGate
  Review these steps to obtain a high-level overview of how to upgrade Oracle Database using Oracle GoldenGate.
- 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, using the Export utility.
- 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.

Starting with Oracle Database 21c, non-CDB Oracle Database upgrades to non-CDB architecture are desupported. Review the upgrade options for this release.

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 19c into an Oracle Database 19c release CDB). Finish converting the non-CDB Oracle Database to a PDB. Then, upgrade the entire CDB, with its PDBs, to Oracle Database 21c.

- 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 21c CDB. Upgrade the plugged-in non-CDB Oracle Database to Oracle Database 21c. 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 Logical 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. Starting with Oracle Database 21c, you can use the Zero Downtime Upgrade feature of Fleet Patching and Provisioning to convert non-container Oracle Database instances (CDBs) to multitenant architecture Oracle Database instances. When you use this feature with Oracle GoldenGate, you can initiate the entire upgrade and conversion as a single operation. • 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.

Overview of Steps for Upgrading Oracle Database Using Oracle GoldenGate

Review these steps to obtain a high-level overview of how to upgrade Oracle Database using Oracle GoldenGate.

Upgrading to the new Oracle Database release using Oracle GoldenGate consists of the following steps.

1. Set up a target database running the earlier Oracle Database software release, using an existing database backup.
2. Upgrade the target database to the new Oracle Database release.
3. Synchronize the target database with the production database.
4. Test your environment in active/live mode.
5. Switch over the application to the target database.
6. Perform comprehensive testing of the new release on the target database, including enabling any features that were installed disabled by default that you plan to use in the new Oracle Database release.
7. Upgrade the production database to the new Oracle Database release.

> **Note:**
>
> For complete details of this procedure, refer to the Oracle GoldenGate documentation.

Related Topics

- Oracle GoldenGate documentation
- Testing a Database 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 Microsoft 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:**
>
> 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.

   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`.

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

   If your existing database is on Microsoft 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, or you can use the Replay Upgrade process.

Migrating from Enterprise Edition to Standard Edition of Oracle Database

Converting from Enterprise Edition to Standard Edition requires exporting and importing data, using the Export utility.

To properly convert from an Enterprise Edition database to a Standard Edition database, you must perform an Export/Import operation. 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

Understand image-based installation to simplify installation and configuration of Oracle Database software.

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. If you install or clone an Oracle Database image, then all Oracle Database options such as Oracle OLAP (`olap`) and Oracle Real Application Testing (`rat`) are enabled by default.

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.
- 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 the New Oracle Home for Upgrading
  To prepare the new Oracle home in a new location, check to see if you must move configuration files, or complete other tasks.
- 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.
- Back Up Oracle Database Before 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.
- Pre-Upgrade Information Check with AutoUpgrade
  To obtain a checklist of tasks you must complete before upgrading an Oracle Database, run the AutoUpgrade utility (`autoupgrade.jar`) in `analyze` mode.
- 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 or Patching
  When you upgrade or patch the database, you install the new Oracle home in a new location (an out-of-place upgrade or patch).
- 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.

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 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

- *Learning Database New Features*

Pre-Upgrade Information Check with AutoUpgrade

To obtain a checklist of tasks you must complete before upgrading an Oracle Database, run the AutoUpgrade utility (`autoupgrade.jar`) in `analyze` mode.

Oracle recommends that you download and run the most recent release of AutoUpgrade in `-analyze` mode before you upgrade Oracle Database. AutoUpgrade can identify issues for you to address before you start your upgrade. In certain cases, AutoUpgrade can also generate scripts that can resolve some issues.

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

Related Topics

- My Oracle Support AutoUpgrade Tool (Doc ID 2485457.1)
- Upgrade your Database – NOW! Mike Dietrich's Oracle Database Upgrade Blog

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.

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 Replay Upgrade Method for Upgrading Oracle Database
  The Replay Upgrade feature upgrades databases to the multitenant architecture by using previously-captured statements to perform the upgrade.
- 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.

Oracle recommends that you download the most recent version of the AutoUpgrade Utility from My Oracle Support Document 2485457.1, and use `autoupgrade.jar` to prepare for and to deploy your upgrade. 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. Starting with Oracle Database 21c, when you have an existing target release CDB, you can use AutoUpgrade to convert a non-CDB Oracle Database to a PDB on the target release CDB during the upgrade.

With AutoUpgrade, when you have an existing target release CDB, you can use AutoUpgrade to convert a non-CDB Oracle Database to a PDB on the target release CDB during the upgrade.

Starting with AutoUpgrade version 22.1, Oracle provides REST APIs that enable you to perform upgrades remotely over SSH using Oracle REST Data Services (ORDS) or Oracle Cloud Infrastructure (OCI) REST API. The ORDS database API is a database management and monitoring REST API embedded into Oracle REST Data Services. The OCI REST API is enabled by configuring the REST Adapter connection to use the OCI Signature Version 1 security policy. You can now use these features to run AutoUpgrade upgrades remotely over HTTPS or HTTP.

The minimum `COMPATIBLE` parameter setting for the source database must be at least 12.2.0. If the `COMPATIBLE` setting is a lower version, then during the conversion and upgrade process, `COMPATIBLE` is set to 12.2.0. During the conversion, the original datafiles are retained. They are not copied to create the new PDB. To enable AutoUpgrade to perform the upgrade, edit the AutoUpgrade configuration file to set the AutoUpgrade parameters `target_version` to the target CDB release, and identify the CDB to which the upgraded database is placed using `target_cdb`. During the conversion and upgrade process, AutoUpgrade uses that information to complete the upgrade to the target CDB.

> **Caution:**
>
> Before you run AutoUpgrade to complete the conversion and upgrade. Oracle strongly recommends that you create a full backup of your source database, and complete thorough testing of the upgrade. There is no option to roll back to the non-CDB Oracle Database state after AutoUpgrade starts this procedure.

The Replay Upgrade Method for Upgrading Oracle Database

The Replay Upgrade feature upgrades databases to the multitenant architecture by using previously-captured statements to perform the upgrade.

The Replay Upgrade feature uses capture tables in the `CDB$ROOT` of the target release CDB Oracle Database. You install the new Oracle Database release, which uses the multitenant architecture. If you are upgrading a CDB, then each PDB in the CDB is upgraded using Replay Upgrade. If you are upgrading individual PDBs, or upgrading non-CDB databases, then you upgrade each database by plugging it in to the new Oracle Database release, and then opening the database. When the database is opened, the source database is upgraded to the new release. If the source database is a non-CDB, then it is converted from a non-CDB to a PDB, and upgrade statements are replayed into the new Oracle Database, updating in the process the data dictionary for the database tables.

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 continue to use DBUA to upgrade multitenant architecture container databases (CDBs), and pluggable databases (PDBs). Starting with Oracle Database 21c, you can no longer use DBCA to upgrade to a non-CDB Oracle Database.

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 AutoUpgrade (`autoupgrade.jar -mode analyze`)

  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.

- Prepare the new Oracle home.
- Perform a backup of the database.

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
  Using Export/Import data migration to move to a new Oracle Database instance can maintain availability, but there are restrictions and tests you should perform.
- Export/Import Benefits for Migrating Data for Oracle Database
  Migrating data when upgrading Oracle Database using Oracle Data Pump Export and 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

Using Export/Import data migration to move to a new Oracle Database instance can maintain availability, but there are restrictions and tests you should perform.

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 for 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 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. Export/Import creates an identical copy of the database, but other factors associated with differences between database releases 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 Oracle Data Pump Export and Import provides benefits that can increase performance.

Using Oracle Data Pump Export and 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 earlier release (current) Oracle Database, and the new release 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. Oracle Data Pump Export and Import

provides flexible data-subsetting capabilities, such as INCLUDE and EXLUDE. By using these capabilities, you can narrow the list of objects that you import. By using the QUERY parameters, you can to filter out rows for a table at export and import time.

- 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 Oracle Data Pump Import enables you to load the new release Oracle Database directly across the network used for your earlier release Oracle Database. By using network-based Oracle 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 or Patching

When you upgrade or patch the database, you install the new Oracle home in a new location (an out-of-place upgrade or patch).

AutoUpgrade performs out-of-place upgrades and patches. This means that the upgrade or the patched Oracle home is in a new Oracle home. 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 or patch process in the out-of-place Oracle home database before replacing your production environment entirely.

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. Because the CDB is the target release, it is already in a new Oracle home for that release. There is no need to choose a new location for installing the target Oracle homes for the PDBs, because the target CDB already has its Oracle home.

Related Topics

- How to Speed Up Your Database and GI Patching

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
  To avoid encountering application startup or invocation problems, Oracle recomends that you perform minimal testing of applications on a test new Oracle Database 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 earlier Oracle Database release, 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

To avoid encountering application startup or invocation problems, Oracle recomends that you perform minimal testing of applications on a test new Oracle Database environment.

Minimal testing for Oracle Database entails moving all or part of an application from the current Oracle Database release to a new release Oracle Database installation, 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 an Oracle Database upgrade on your production workload before actually upgrading the production database.
- SQL Performance Analyzer
  To forecast the impact of Oracle Database system changes on a SQL workload, use the SQL Performance Analyzer.
- Use SQL Plan Management to Test SQL Execution Plans After Upgrade
  To avoid performance regressions after an Oracle Database upgrade, learn how to carry out SQL plan management tests.

Database Replay and Performance Testing

Use the Database Replay feature to perform real-world testing of an Oracle 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.

Related Topics

- Introduction to Database Replay
- Managing the Automatic Workload Repository

SQL Performance Analyzer

To forecast the impact of Oracle Database system changes on a SQL workload, use the SQL Performance Analyzer.

SQL Performance Analyzer enables you to evaluate the effect of an Oracle Database 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.

Related Topics

- Introduction to SQL Performance Analyzer

Use SQL Plan Management to Test SQL Execution Plans After Upgrade

To avoid performance regressions after an Oracle Database upgrade, learn how to carry out SQL plan management tests.

- Why Perform SQL Plan Management?
  To prevent users from encountering performance regressions after an Oracle Database upgrade, carry out SQL plan management.
- 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 an earlier release to the latest release of Oracle Database.
- Bulk Load a SQL Management Base with a SQL Tuning Set (STS)
  Bulk loading of execution plans or SQL plan baselines is useful to load historic plans from the Automatic Workload Repository.
- Unpack Existing SQL Plan Baselines from a Staging Table
  Test your critical SQL queries and execution plans by using `DBMS_SPM.LOAD_PLAN_FROM_CURSOR_CACHE` to create a staging table that you can migrate for testing.

Why Perform SQL Plan Management?

To prevent users from encountering performance regressions after an Oracle Database upgrade, carry out SQL plan management.

An Oracle 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 can 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 upgraded Oracle Database optimizer determines that a different plan can result in better performance, then the new plan is queued for verification. The new plan is not used until it has been confirmed to have comparable or better performance than the current plan.

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 an earlier release to the latest release of Oracle Database.

The cursor cache is a shared SQL area. 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.

Related Topics

- Loading Plans from the Shared SQL Area

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

Bulk loading of execution plans or SQL plan baselines is useful to load historic plans from the Automatic Workload Repository.

Bulk loading of execution plans or SQL plan baselines may be done with a SQL Tuning Set. This is useful when you want to load historic plans from your earlier Oracle Database 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.

Related Topics

- Loading Plans from a SQL Tuning Set

Unpack Existing SQL Plan Baselines from a Staging Table

Test your critical SQL queries and execution plans by using `DBMS_SPM.LOAD_PLAN_FROM_CURSOR_CACHE` to create a staging table that you can migrate for testing.

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 new Oracle Database release 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 Oracle Data Pump.
5. Transfer this flat file to the target system.
6. Import the staging table from the flat file using Oracle 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.

Related Topics

- Loading Plans from a Staging Table
- Overview of Oracle 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 earlier Oracle Database release, and on the test database that you upgraded to the new Oracle Database release.

When you perform your plan tests:

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

To verify that your existing applications operate properly with the new Oracle Database release:

- 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.

Related Topics

- 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, Flashback Database can't be used to revert an unplug-plug upgrade or PDB conversion. For unplug-plug upgrades, rely on other fallback strategies, such as an RMAN backup.

If you use AutoUpgrade, then Oracle recommends that you specify `target_pdb_copy_option=file_name_convert`, in the AutoUpgrade configuration file, where `file_name_convert` is a convert pattern prefixed to the data files. When you do that, AutoUpgrade directs the database to create copies of the data files before plugging in the database. Choosing to use this method enables you to use the original database as a fallback. However, be aware that when you create data file copies, the upgrade requires additional disk space and extra time.

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*.

In addition, to ensure that you are prepared for a downgrade, review the downgrade chapter and complete any preparation steps you may need to prepare for your release.

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.
```

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.

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

   When installation of Oracle Database software has completed successfully, click **Exit** to close Oracle Universal Installer.
4. Run AutoUpgrade with the `preupgrade` parameter, run in `analyze` mode. AutoUpgrade can automatically fix many issues, and list other issues in the `prefixups` file it generates, which you can fix manually before the upgrade.
5. Run AutoUpgrade in Deploy mode.

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).
- Upgrades and Transparent Data Encryption
  To upgrade databases using TDE, provide AutoUpgrade with TDE passwords either by using the `-load_password` command line option, or by specifying an external password store.
- Recommendations for Oracle Net Services When Upgrading Oracle Database
  You must ensure that the listener is running in your new release Oracle home.
- When You Must Disable Oracle Database Vault
  You may need to disable Oracle Database Vault to perform upgrade tasks or correct erroneous configurations.
- Create or Migrate Your Password File with ORAPWD
  Review if you have `REMOTE_LOGIN_PASSWORDFILE` set.

- Understanding Password Case Sensitivity and Upgrades
  By default, Oracle Database 12c Release 2 (12.2) and later releases use Exclusive Mode authentication protocols. 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.
- Resource and Password Parameter Updates for STIG and CIS Profiles
  Starting with Oracle Database 21c, the upgrade configures Oracle Recommended Profiles, which includes updating an already existing STIG profile, and installing a CIS profile as part of the upgrade.
- 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.
- Moving Operating System Audit Records into the Unified Audit Trail
  Audit records that have been written to the spillover audit files can be moved to the unified audit trail database table.
- 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.
- Back Up Very Large Databases Before Using AutoUpgrade
  If you use partial offline backups with very large databases, then to minimize downtime in the event you need to downgrade your database, check your tablespaces and ensure that all tablespaces required for recovery are backed up.

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 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

Upgrades and Transparent Data Encryption

To upgrade databases using TDE, provide AutoUpgrade with TDE passwords either by using the `-load_password` command line option, or by specifying an external password store.

Starting with AutoUpgrade version 22.1, you can choose either to provide Transparent Data Encryption (TDE) passwords at the command line during the upgrade to access the source keystores, and have AutoUpgrade create new external keystore on the target system in a location that you choose, or you can specify that AutoUpgrade should access an existing secure external password store (SEPS) that contains the TDE passwords.

Provide TDE Passwords At the Command Line Using Password Initialization and Storage

If you have the TDE passwords for the databases that you want to upgrade, then you can provide those passwords to AutoUpgrade at the command line. AutoUpgrade creates an external key manager generated and maintained by AutoUpgrade. With this configuration, AutoUpgrade supports unmanned or automated operations of TDE-enabled databases. As the upgrade runs, AutoUpgrade can open each source database keystore without prompting for the keystore password, and enroll the target database into the TDE external keystore for key management, so that the target database can start automatically.

1. Before running AutoUpgrade, you add the global parameter `global.keystore` to the configuration file that you use with a database that uses TDE, and specify a secure path to the location of the keystore that you want created for the upgraded database. This path should be different from any other file path you specify in AutoUpgrade, so that the keystore is not in any log file location.
2. When you run AutoUpgrade in Deploy mode, you must run it using the `-config` command line parameter, and with the `-load_password` parameter.
3. Before AutoUpgrade starts the database upgrades, AutoUpgrade prompts you to provide the TDE passwords for each database specified in the configuration file that uses TDE. These passwords are used only to access the source release TDE keystores, and to write the TDE passwords to the new target external keystore. No passwords are written to SQL*Plus execution plans during the upgrade. After AutoUpgrade no longer requires the TDE passwords, these passwords are purged from memory. No log records are kept of the passwords.

Note the following security features of the password initialization and storage option:

- You specify the TDE passwords as AutoUpgrade starts to deploy; they are not included in the configuration file.
- AutoUpgrade prompts you to provide the TDE password for each source database specified in your configuration file that contains a TDE keystore.
- AutoUpgrade performs no password logging in any files written by AutoUpgrade during the upgrade. Instead, AutoUpgrade records that the `load_password` command line option was used during the Deploy.

- As AutoUpgrade runs, it places TDE passwords entered at the command line into secure Java KeyStore objects.
- After the TDE password for an Oracle Database keystore is used for access, and the target database is enrolled into the TDE external keystore for key management, AutoUpgrade clears the Java KeyStore objects containing the password, so that these passwords are no longer in memory.
- AutoUpgrade does not include the keystore file in the zip file that AutoUpgrade generates during the upgrade.
- If the upgrade is a non-CDB or an Unplug-Plug PDB upgrade, then the XML manifest file created by AutoUpgrade for databases undergoing a Non-CDB to PDB or Unplug/Plug upgrade contains TDE encryption keys. This file is also excluded in the zip file generated by AutoUpgrade.

Provide TDE Passwords Using an AUTO LOGIN keystore

If you choose to use an existing external keystore to provide AutoUpgrade with passwords for TDE, then you must perform a one-time setup of an `AUTO LOGIN` keystore, so that the database can be shut down and restarted without requiring DBA intervention.

To review your existing have an Oracle Wallet value specified, enter the following command:

```
SQL> show parameter WALLET_ROOT;
```

With AutoUpgrade 22.1 and later, copying the `sqlnet.ora file` to the new Oracle release is no longer required. If you choose to use a secure external password keystore, then Oracle recommends that you use the `WALLET_ROOT` static initialization parameter and `TDE_CONFIGURATION` dynamic initialization parameter.

You can specify in your AutoUpgrade configuration file to have the keystore location changed during the upgrade. Alternatively, you can complete this task manually after the upgrade. In either case, ensure that a recent backup has been made of the keystore before you start the upgrade.

If you complete this task manually, then after the upgrade, before you can configure keystores and begin to encrypt data, you must perform a one-time configuration using the `WALLET_ROOT` and `TDE_CONFIGURATION` parameters to designate the location and type of keystores that you plan to create.

The `WALLET_ROOT` parameter specifies the keystore directory location. Before you set `WALLET_ROOT`, ensure that you have an existing directory that you can use to store keystores. (Typically, this directory is called `wallet`.)

The `TDE_CONFIGURATION` parameter specifies the type of keystore (software keystore, hardware keystore, or Oracle Key Vault keystore). If you omit the `TDE_CONFIGURATION` parameter, then Oracle Database uses the `sqlnet.ora` file settings. After you set the type of keystore using `TDE_CONFIGURATION`, when you create the keystore, Oracle Database creates a directory within the `WALLET_ROOT` location for the keystore type. For example, if you set `TDE_CONFIGURATION` to `FILE`, for Transparent Data Encryption keystores, then Oracle Database creates a directory named `tde` (lower case) within the wallet directory. If you want to migrate from one keystore type to another, then you must first set `TDE_CONFIGURATION` parameter to the keystore type that you want to use, and then use the `ADMINISTER KEY MANAGEMENT` statement to perform the migration. For example, you can migrate from a hardware security module (HSM) keystore to a TDE keystore.

The KEYSTORE_MODE column of the V$ENCRYPTION_WALLET dynamic view shows whether united mode or isolated mode has been configured.

> **Note:**
>
> In previous releases, the SQLNET.ENCRYPTION_WALLET_LOCATION parameter was used to define the keystore directory location. This parameter has been deprecated. Oracle recommends that you use the WALLET_ROOT static initialization parameter and TDE_CONFIGURATION dynamic initialization parameter instead. You can use the AutoUpgrade utility to perform this update for you during the upgrade.

Related Topics

- Databases are Fun: AutoUpgrade and Transparent Data Encryption (TDE)
- Managing the Keystore and the Primary Encryption Key
- 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.

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.

For Oracle RAC database upgrades, the listener normally is migrated during the Grid Infrastructure upgrade. 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.

Related Topics

- *Oracle Database Net Services Reference*

When You Must Disable Oracle Database Vault

You may need to disable Oracle Database Vault to perform upgrade tasks or correct erroneous configurations.

You can reenable Oracle Database Vault after you complete the corrective tasks.

The following situations require you to disable Oracle Database Vault:

- You must install any of the Oracle Database optional products or features, such as Oracle Spatial, by using Database Configuration Assistant (DBCA).

- If you did not configure backup DV_OWNER and DV_ACCTMGR accounts when you registered Oracle Database Vault, and these accounts are inadvertently locked or their passwords forgotten. Note that if your site only has one DV_OWNER user and this user has lost his or her password, you will be unable to disable Oracle Database Vault. However, if your site's only DV_ACCTMGR user has lost the password, you can disable Database Vault. As a best practice, you should grant the DV_OWNER and DV_ACCTMGR roles to new or existing user accounts, and use the Database Vault Owner and Account Manager accounts that you created when you registered Database Vault as back-up accounts.
- If you want to register Oracle Internet Directory (OID) using Oracle Database Configuration Assistant (DBCA).
- If Oracle Database Vault is enabled and you are upgrading an entire CDB, then use one of the following methods:
  - **CDB upgrade method 1:** Temporarily grant the DV_PATCH_ADMIN to user SYS commonly by logging into the root container as a common user with the DV_OWNER role, and then issuing the GRANT DV_PATCH_ADMIN TO SYS CONTAINER=ALL statement. Oracle Database Vault controls will be in the same state as it was before the upgrade. When the upgrade is complete, log into the root container as the DV_OWNER user and revoke the DV_PATCH_ADMIN role from SYS by issuing the REVOKE DV_PATCH_ADMIN FROM SYS CONTAINER=ALL statement.
  - **CDB upgrade method 2:** Log into each container as a user who has the DV_OWNER role and then execute the DBMS_MACADM.DISABLE_DV procedure. You must first disable the PDBs (in any order) and then after that, disable the root container last. If you are upgrading only one PDB, then you can disable Oracle Database Vault in that PDB only. After you have completed the upgrade, you can enable Oracle Database Vault by logging into each container as the DV_OWNER user and then executing the DVSYS.DBMS_MACADM.ENABLE_DV procedure. The order of enabling Oracle Database Vault must be the root container first and PDBs afterward. You can enable the PDBs in any order, but the root container must be enabled first.

> **Note:**
>
> Be aware that if you disable Oracle Database Vault, the privileges that were revoked from existing users and roles during the Oracle Database Vault configuration remain in effect.

Related Topics

- Verifying That Database Vault Is Configured and Enabled
- Backup Oracle Database Vault Accounts
- Privileges That Are Revoked from Existing Users and Roles

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`

Related Topics

- ORAPWD Syntax and Command Line Argument Descriptions

Understanding Password Case Sensitivity and Upgrades

By default, Oracle Database 12c Release 2 (12.2) and later releases use Exclusive Mode authentication protocols. 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.

> **Note:**
>
> Starting with Oracle Database 21c, the `SEC_CASE_SENSITIVE_LOGON` parameter is desupported. You must use a case-sensitive password version. If a user with only a `10G` password version is upgraded to Oracle Database 21c, then that user account is locked, until an administrator resets the password.

In previous Oracle Database releases, you could configure the authentication protocol so that it allows case-insensitive password-based authentication by setting `SEC_CASE_SENSITIVE_LOGON=FALSE`. Starting with Oracle Database 12c release 2 (12.2), the default password-based authentication protocol configuration excluded 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 12c 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.

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 11g 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.

Update Accounts Using Case-Insensitive Versions

If you have user accounts that have only the case-insensitive `10G` password version, then 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.

Related Topics

- *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.

> **Note:**
>
> Starting with Oracle Database 21c, the `SEC_CASE_SENSITIVE_LOGON` parameter is desupported. You must use a case-sensitive password version.

If you do not want user accounts authenticated with case-insensitive password versions to be locked out of the database after an upgrade, then before the upgrade, 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*

Resource and Password Parameter Updates for STIG and CIS Profiles

Starting with Oracle Database 21c, the upgrade configures Oracle Recommended Profiles, which includes updating an already existing STIG profile, and installing a CIS profile as part of the upgrade.

A **profile** is a collection of attributes that apply to a user. It enables a single point of reference for any of multiple users that share those exact attributes.

During Oracle Database upgrades, the Oracle Supplied Profile `ORA_STIG_PROFILE` user profile is updated in accordance with the most recent system configuration baselines specified by the US Department of Defense Systems Agency (DISA) Security Technical Implementation Guides (STIG) baselines. This update overwrites any password and resource limits that you may have set previously in the `ORA_STIG_PROFILE` user profile. In addition, a new profile is added, `ORA_CIS_PROFILE`, which complies with the most recent Center of Internet Security (CIS) baseline updates available to Oracle at the time of the software release. These two profiles are designated **Oracle Recommended Profiles**. These profiles differ from a standard `DEFAULT` profile, because they are based on the STIG and CIS baselines.

The profiles `ORA_STIG_PROFILE` and `ORA_CIS_PROFILE` are created as `LOCAL` profiles, and the clause `CONTAINER=CURRENT` clause is used. However, to enhance the security of the profiles that Oracle provides, only the `SYS` user has permissions to modify these files.

If there are users associated to `ORA_STIG_PROFILE`, then the following parameters for these users are made stricter after the upgrade:

- `PASSWORD_LIFE_TIME`, which is changed to `35`.
- `PASSWORD_REUSE_TIME`, which is changed to `175`.
- `PASSWORD_GRACE_TIME`, which is changed to `0`.

For more information about using Oracle Recommended Profiles, refer to *Oracle Database Security Guide*.

Related Topics

- Managing Resources with Profiles

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`.

> **Note:**
>
> If you are performing a non-CDB to PDB conversion, then using read-only tablespaces is not a valid fallback option. During a non-CDB to PDB conversion, tablespaces must be online during conversion, because each data file header requires changes during the upgrade.

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 Advanced 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*

Moving Operating System Audit Records into the Unified Audit Trail

Audit records that have been written to the spillover audit files can be moved to the unified audit trail database table.

When the database is not writable (such as during database mounts), if the database is closed, or if it is read-only, then Oracle Database writes the audit records to these external files. The default location for these external files is the `$ORACLE_BASE/audit/$ORACLE_SID` directory.

You can load the files into the database by running the `DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES` procedure. Be aware that if you are moving a large number of operating system audit records in the external files, performance may be affected.

To move the audit records in these files to the `AUDSYS` schema audit table when the database is writable:

1. Log into the CDB root as a user who has been granted the `AUDIT_ADMIN` role.

   Before you can upgrade to the current release or Oracle Database, you must execute the `DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES` procedure from the CDB root to avoid losing operating system spillover files during the upgrade process.

   For example:

   ```
   CONNECT c##aud_admin
   Enter password: password
   Connected.
   ```

2. Ensure that the database is open and writable.

   To find if the database is open and writable, query the `V$DATABASE` view.

   ```
   SELECT NAME, OPEN_MODE FROM V$DATABASE;

   NAME            OPEN_MODE
   --------------- ----------
   HRPDB           READ WRITE
   ```

You can run the `show pdbs` command to find information about PDBs associated with the current instance.

3. Run the `DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES` procedure.

```
EXEC DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES;
```

4. If you want to load individual PDB audit records, then log in to each PDB and run the `DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES` procedure again.

The audit records are loaded into the `AUDSYS` schema audit table immediately, and then deleted from the `$ORACLE_BASE/audit/$ORACLE_SID` directory.

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

Back Up Very Large Databases Before Using AutoUpgrade

If you use partial offline backups with very large databases, then to minimize downtime in the event you need to downgrade your database, check your tablespaces and ensure that all tablespaces required for recovery are backed up.

If you are using the AutoUpgrade utility for upgrading databases where you have selected partial offline backups as your backup option, then check that all tablespaces that are required for upgrade are in `READ WRITE` mode, and only after you are sure you have identified all required tablespaces for backup, change the status of all required tables before you take an `OFFLINE` backup of the tablespaces you require for recovery before you run AutoUpgrade.

The reasons for this guideline are as follows:

During an AutoUpgrade operation, other tablespaces besides `SYSTEM`, `SYSAUX` and `UNDO` may need to be maintained in `READ WRITE` status for the upgrade. Some of the reasons for this requirement during an upgrade can include:

- Tablespaces that contain dictionary objects
- Tablespaces that are the default tablespace for Oracle-maintained users
- Tablespaces that are the default tablespace for the database

AutoUpgrade detects if all tablespaces needed for the upgrade are in `READ WRITE` status.

When there are tablespaces that must be changed to `READ WRITE` mode for the upgrade, then:

- During the `PRECHECKS` processing mode, AutoUpgrade detects tables in `READ ONLY` status as an issue.
- During the `FIXUP` processing mode, AutoUpgrade performs an automatic fixup to update to `READ WRITE` mode any tablespaces that it detects in `READ ONLY` mode that must be in `READ WRITE` mode.

If there are any tablespaces required for upgrade that AutoUpgrade changes from `READ ONLY` mode to `READ WRITE` mode, and these tablespaces were not included in your backup before starting AutoUpgrade, then your recovery strategy is at risk. To ensure that your backup is valid for recovery, you must take your `OFFLINE` backup only after you are sure which tablespaces must be backed up.

To ensure that your partial offline backup contains backups for all tablespaces modified during the upgrade, complete this procedure:

1. Put all tablespaces in `READ ONLY` mode, except for `SYSTEM`, `SYSUX` and `UNDO` and those tablespaces that you know must be in `READ WRITE`.
2. Run the this query for pivot users:

```
(SELECT username
FROM dba_users
WHERE user_id in (
  SELECT schema# FROM sys.registry$
    WHERE namespace = 'SERVER'
    UNION
    SELECT schema# FROM sys.registry$schemas
    WHERE namespace = 'SERVER'
    UNION
    SELECT user# FROM sys.user$
```

```
  WHERE type#=1 AND bitand(spare1,256)=256))
SELECT tablespace_name
FROM dba_tablespaces
WHERE status <>'ONLINE' and tablespace_name IN
(
SELECT property_value
FROM database_properties
  WHERE property_name = 'DEFAULT_PERMANENT_TABLESPACE'
  UNION
  SELECT default_tablespace
  FROM dba_users
  WHERE username IN (SELECT username FROM pivot_users)
  UNION
  SELECT tablespace_name
  FROM dba_segments
  WHERE owner IN (SELECT username FROM pivot_users)
  UNION
  SELECT t.name
  FROM modeltab$ m,  ts$ t, sys_objects s
  WHERE m.obj#=s.object_id and s.ts_number=t.ts#
  )'
```

The next step you take depends on the result of the query:

- If the query returns no rows, then it means that backing up `SYSTEM`, `SYSAUX` and `UNDO`, as well as those tables you specifically know must be in `READ WRITE`, is sufficient to complete a partial offline backup.
- If the query return rows in tablespaces, then to complete a partial offline backup, you must place these additional tablespaces in `READ WRITE` mode.

3. When you have completed identifying and placing all required tablespaces in `READ WRITE` mode, take your partial offline backup of those tablespaces. Also back up `SYSTEM`, `SYSAUX` and `UNDO`.redo logs, control files and any other files that you consider relevant for the restore/recovery procedure in case they are needed.
4. Run AutoUpgrade in `ANALYZE` mode. Review the output, and ensure that AutoUpgrade identifies no additional tablespaces reported as `READ ONLY` that must be put in `READ WRITE`.

(Optional) Enter the result of the procedure here.

Preparing the New Oracle Home for Upgrading

To prepare the new Oracle home in a new location, check to see if you must move configuration files, or complete other tasks.

After backing up the database that you want to upgrade, if you are not using a Read-Only Oracle home, then prepare the new Oracle home in a new location, and 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 a Replay Upgrade, the AutoUpgrade Utility, or DBUA for your upgrade, or you have a Read-Only Oracle home, 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 one of the following commands:

```
CREATE pfile FROM spfile;
```

You can also create the parameter file by using the following command, where */path/to/pfile/* is the path to the new Oracle home, and *pfile_name* is the name of the parameter file:

```
create pfile[='/path/to/pfile/pfile_name.ora/
```

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 earlier release 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.
- Microsoft 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.

AutoUpgrade run with the `-preupgrade` parameter in `analyze` mode displays any deprecated parameters and desupported parameters it finds in the `upgrade.xml` file that it generates.

Adjust the values of the initialization parameters to at least the minimum values indicated in `upgrade.xml`

Ensure 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 AutoUpgrade or Replay Upgrade, 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, keystores, 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.
- Microsoft Windows virtual accounts, which are managed local accounts that use computer credentials to access network resources, require additional attention during Oracle Database installation. Ensure that you create a virtual account for the Oracle Home User.

> **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.

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=21`.

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 `dbsales`, 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:

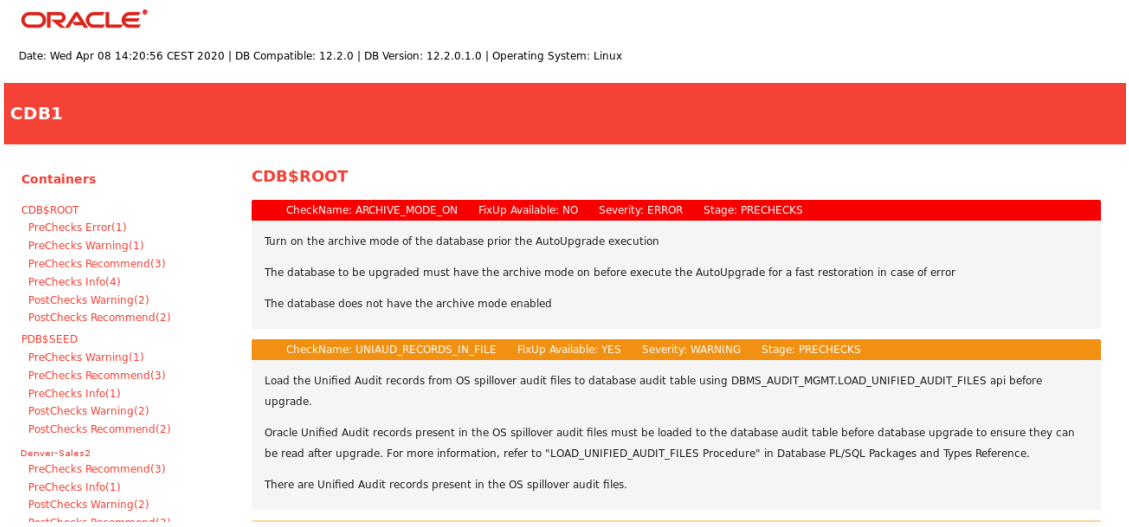
*/log-path/sid/job-number/*prechecks

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.

- Preparing for Upgrades of Databases with Oracle Database Vault
  If the Oracle Database you plan to upgrade uses Oracle Database Vault, then you must disable Oracle Database Vault before starting the upgrade.

Preparing for Upgrades of Databases with Oracle Database Vault

If the Oracle Database you plan to upgrade uses Oracle Database Vault, then you must disable Oracle Database Vault before starting the upgrade.

During the upgrade process, if your source Oracle Database uses Oracle Database Vault, then you must first disable Oracle Database Vault before you start the upgrade.

You have two options you can use:

1. Use a manual procedure: Log on as the common Database Vault (DV) administrator in the `CDB$ROOT` and grant the `DV_PATCH_ADMIN` role to `SYS`, or log in and disable Oracle Database Vault on every container. Procedures vary slightly, depending on your upgrade scenario. This procedure is described in My Oracle Support, "Requirement for Upgrading Database with Database Vault (Doc ID 2757126.1)".
2. Download the latest AutoUpgrade Jar file, and perform the procedure described here.

With either option, when you run AutoUpgrade in Analyze mode, it detects that Oracle Database Vault is enabled, and indicates in its report that you must ensure the prerequisites for Oracle Database Vault and upgrade are met.

Example 2-7 AutoUpgrade Procedure for Databases Using Oracle Database Vault

When you use AutoUpgrade, and your database is configured with Oracle Database Vault, the upgrade procedure is as follows:

1. Disable Oracle Database 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
- Requirement for Upgrading Database with Database Vault (Doc ID 2757126.1)
- AutoUpgrade Tool (Doc ID 2485457.1)

Back Up Oracle Database Before 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 strongly recommends that you create a backup of the Oracle software and databases. For Oracle software running on Microsoft Windows operating systems, you must also take a backup of the Windows registry. On Microsoft Windows, 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.

Before you cleanly shut down the database, you must run AutoUpgrade using the `preupgrade` parameter. To minimize downtime, you can perform an online backup, or create a guaranteed restore point.

1. Sign on to Oracle RMAN:

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

2. Run the following RMAN commands:

   ```
   RUN
   {
       ALLOCATE CHANNEL chan_name TYPE DISK;
       sql 'ALTER SYSTEM ARCHIVE LOG CURRENT';
       BACKUP DATABASE FORMAT '/tmp/db%U' TAG before_upgrade
       PLUS ARCHIVELOG FORMAT '/tmp/arch%U' TAG before_upgrade;
       BACKUP CURRENT CONTROLFILE FORMAT '/tmp/ctl%U' TAG before_upgrade;
   }
   ```

   > **Caution:**
   >
   > You must ensure that no other RMAN backup runs during this backup. If another RMAN command backs up and removes archive log, then this backup could be unrecoverable, because the other RMAN command has removed the archive logs.

Related Topics

- About Online Backups and Backup Mode
- Using Flashback Database and Restore Points
- Backing Up the Database

3

Upgrading Databases with Oracle Data Guard Standbys

When you upgrade a database to a new release that uses one or more Oracle Data Guard Standby databases, you use the redo logs from the primary database.

This scenario assumes you are using Oracle Data Guard broker.

- Preparing for Upgrades of Databases Using Oracle Data Guard
  In this scenario, you perform your upgrade using Oracle Data Guard with Data Guard broker, and move the Data Guard broker configuration files before starting your upgrade.
- Manual Non-CDB to PDB Conversion and Upgrade with Data Guard Standby
  Upgrades with Oracle Data Guard where you want to uprade a non-CDB to a PDB, and reuse the non-CDB stadnby datafiles require a special procedure.
- Before You Patch or Upgrade the Oracle Database Software
  Before you patch or upgrade your Oracle Database software, review the prerequisites for different use case scenarios.
- Recovering After the NOLOGGING Clause Is Specified
  Some SQL statements allow you to specify a `NOLOGGING` clause so that the operation is not logged in the online redo log file.
- Enable an Appropriate Logging Mode
  As part of preparing the primary database for standby database creation, you must enable a logging mode appropriate to the way you plan to use the Oracle Data Guard configuration.
- Creating a Physical Standby Task 1: Create a Backup Copy of the Primary Database Data Files
  You can use any backup copy of the primary database to create the physical standby database, as long as you have the necessary archived redo log files to completely recover the database.
- Creating a Physical Standby Task 2: Create a Control File for the Standby Database
  Create the control file for the standby database. The primary database does not have to be open, but it must at least be mounted.
- Creating a Physical Standby Task 3: Create a Parameter File for the Standby Database
  Create a parameter file (`PFILE`) from the server parameter file (`SPFILE`) used by the primary database.
- Upgrading Oracle Database with a Physical Standby Database in Place
  These steps show how to upgrade to Oracle Database when a physical standby database is present in the configuration.
- Creating a Physical Standby Task 4: Copy Files from the Primary System to the Standby System
  Ensure that all required directories are created. Use an operating system copy utility to copy binary files from the primary system to their correct locations on the standby system.

- Creating a Physical Standby Task 5: Set Up the Environment to Support the Standby Database
  Set up the environment by creating a Windows-based service, a password file, and an SPFILE, and then setting up the Oracle Net environment.
- Creating a Physical Standby Task 6: Start the Physical Standby Database
  These are the steps to start the physical standby database and Redo Apply.
- Creating a Physical Standby Task 7: Verify the Physical Standby Database Is Performing Properly
  After you create the physical standby database and set up redo transport services, you may want to verify database modifications are being successfully transmitted from the primary database to the standby database.

Preparing for Upgrades of Databases Using Oracle Data Guard

In this scenario, you perform your upgrade using Oracle Data Guard with Data Guard broker, and 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, if you are not using Oracle Automatic Storage Management (Oracle ASM) for storage, then 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.

   > **Note:**
   >
   > Prior to Oracle Database 21c, the default ORACLE_HOME layout combined ORACLE_HOME, ORACLE_BASE_HOME and ORACLE_BASE_CONFIG into a single location. Starting with Oracle Database 21c, the only available configuration is a read-only ORACLE_HOME where ORACLE_BASE_HOME and ORACLE_BASE_CONFIG are located separately from ORACLE_HOME. Files such as the Oracle Data Guard Files, which were previously located in the folder dbs, are now located in ORACLE_BASE_CONFIG/dbs.

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.

See a Demonstration of how to upgrade with AutoUpgrade and Oracle Data Guard

Daniel Overby Hansen provides you with an overview of how you can use AutoUpgrade to perform upgrades of Oracle Databases using Oracle Data Guard standby databases.

How to Upgrade with AutoUpgrade and Data Guard

Related Topics

- Renaming the Broker Configuration Files

Manual Non-CDB to PDB Conversion and Upgrade with Data Guard Standby

Upgrades with Oracle Data Guard where you want to uprade a non-CDB to a PDB, and reuse the non-CDB stadnby datafiles require a special procedure.

When you have an Oracle Data Guard deployment on your non-CDB source Oracle Database configuration, and you want to reuse the source standby database files on your target PDB after you plug the non-CDB in as a PDB into the primary database of an Oracle Data Guard configuration, there are particular steps required that you must perform manually. For the current release of AutoUpgrade, this functionality is not supported.

For instructions on how to complete an upgrade with this scenario, refer to *Reusing the Source Standby Database Files When Plugging a non-CDB as a PDB into the Primary Database of a Data Guard Configuration (Doc ID 2273304.1)*

Related Topics

- Reusing the Source Standby Database Files When Plugging a non-CDB as a PDB into the Primary Database of a Data Guard Configuration (Doc ID 2273304.1)

Before You Patch or Upgrade the Oracle Database Software

Before you patch or upgrade your Oracle Database software, review the prerequisites for different use case scenarios.

- If you are using the Oracle Data Guard broker to manage your configuration, follow the instructions in *Oracle Data Guard Broker*

- Use procedures described in these topics in conjunction with other upgrade procedures and guidelines provided in *Oracle Database Upgrade Guide*.
- Check for `NOLOGGING` operations. If `NOLOGGING` operations have been performed then you must update the standby database.

  See Recovering After the NOLOGGING Clause Is Specified
- Make note of any tablespaces or data files that need recovery due to `OFFLINE IMMEDIATE`. Before starting an upgrade, tablespaces or data files should be recovered, and either online or offline.
- In an Oracle Data Guard configuration, all physical and snapshot standby databases must use a copy of the password file from the primary database. Password file changes done on the primary database are automatically propagated to standby databases. Password file changes are events such as when an administrative privilege (`SYSDG`, `SYSOPER`, `SYSDBA`, and so on) is granted or revoked, and when the password of any user with administrative privileges is changed.

  Far sync instances are an exception to the automatic updating feature. Updated password files must still be manually copied to far sync instances, because far sync instances receive redo, but do not apply it. When a password file is manually updated at a far sync instance, the redo containing the same password changes from the primary database is automatically propagated to any standby databases that are set up to receive redo from that far sync instance. The password file is updated on the standby when the redo is applied.

> **Note:**
>
> If there are cascaded standbys in your configuration, then those cascaded standbys must follow the same rules as any other standby, but should be shut down last, and restarted in the new home first.

Related Topics

- Oracle Data Guard Broker Upgrading and Downgrading

Recovering After the NOLOGGING Clause Is Specified

Some SQL statements allow you to specify a `NOLOGGING` clause so that the operation is not logged in the online redo log file.

In actuality, when you specify `NOLOGGING`, a redo record is still written to the online redo log file, but there is no data associated with the record. This specification can result in log application or data access errors at the standby site. Manual recovery might be required to resume applying log files. Depending on whether you have a logical standby or physical standby, you can avoid these errors by doing the following:

- Logical standbys

  Specify the `FORCE LOGGING` clause in the `CREATE DATABASE` or `ALTER DATABASE` statements.
- Physical standbys

  Specify a logging mode that is appropriate to the way in which you plan to use your Data Guard configuration.

  See Enable an Appropriate Logging Mode.

You can see the current logging mode in the `V$DATABASE.FORCE_LOGGING` column (for CDBs), or the `DBA_PDBS.FORCE_LOGGING` column (for PDBs).

Enable an Appropriate Logging Mode

As part of preparing the primary database for standby database creation, you must enable a logging mode appropriate to the way you plan to use the Oracle Data Guard configuration.

The default logging mode of a database that is not part of an Oracle Data Guard configuration allows certain data loading operations to be performed in a nonlogged manner. This default mode is not appropriate to a database with a standby, because it leads to the loaded data being missing from the standby, which requires manual intervention to fix.

In addition to the default logging mode, there are three other modes that are appropriate for a primary database:

- `FORCE LOGGING` mode prevents any load operation from being performed in a nonlogged manner. This mode can slow down the load process, because the loaded data must be copied into the redo logs. `FORCE LOGGING` mode is enabled using the following command:

  ```
  SQL> ALTER DATABASE FORCE LOGGING;
  ```

- `STANDBY NOLOGGING FOR DATA AVAILABILITY` mode causes the load operation to send the loaded data to each standby through its own connection to the standby. The commit is delayed until all the standbys have applied the data as part of running managed recovery in an Active Data Guard environment. It is enabled with the following command:

  ```
  SQL> ALTER DATABASE SET STANDBY NOLOGGING FOR DATA AVAILABILITY;
  ```

- `STANDBY NOLOGGING FOR LOAD PERFORMANCE` is similar to the previous mode except that the loading process can stop sending the data to the standbys if the network cannot keep up with the speed at which data is being loaded to the primary. In this mode it is possible that the standbys may have missing data, but each standby automatically fetches the data from the primary as a normal part of running managed recovery in an Active Data Guard environment. It is enabled with the following command:

  ```
  SQL> ALTER DATABASE SET STANDBY NOLOGGING FOR LOAD PERFORMANCE;
  ```

When you issue any of these statements, the primary database must at least be mounted (and it can also be open). The statement can take a considerable amount of time to complete, because it waits for all unlogged direct write I/O to finish.

> **Note:**
>
> When you enable `STANDBY NOLOGGING FOR DATA AVAILABILITY` or `STANDBY NOLOGGING FOR LOAD PERFORMANCE` on the primary database, any standbys that are using multi-instance redo apply functionality will stop applying redo with the error `ORA-10892`. You must first restart redo apply and allow the affected standbys to progress past the NOLOGGING operation period and then enable multi-instance redo apply.
>
> Automatic Correction of Non-logged Blocks at a Data Guard Standby Database" is only available on:
>
> - Oracle Database Appliance
> - Exadata
> - Exadata Cloud Service
> - Database Cloud Service Enterprise Edition - Extreme Performance

Related Topics

- Specifying FORCE LOGGING Mode

> **See Also:**
>
> *Oracle Database Administrator's Guide*For more information about the ramifications of specifying `FORCE LOGGING` mode

Creating a Physical Standby Task 1: Create a Backup Copy of the Primary Database Data Files

You can use any backup copy of the primary database to create the physical standby database, as long as you have the necessary archived redo log files to completely recover the database.

You can use any backup copy of the primary database to create the physical standby database, as long as you have the necessary archived redo log files to completely recover the database. Oracle recommends that you use the Recovery Manager utility (`RMAN`).

Related Topics

- Backing Up the Database, *Oracle Database Backup and Recovery User's Guide*

Creating a Physical Standby Task 2: Create a Control File for the Standby Database

Create the control file for the standby database. The primary database does not have to be open, but it must at least be mounted.

You must create a control file for the standby database. You cannot use a single control file for both the primary and standby databases. They each must have their own file.

Example 3-1 Creating the Control File for the Standby Database

The `ALTER DATABASE` command designates the database that you want to operate in the standby role. In this example, that standby database is named `boston`:

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/tmp/boston.ctl';
```

> **Note:**
>
> If a control file backup is taken on the primary, and restored on a standby (or vice-versa), then the location of the snapshot control file on the restored system is configured to be the default. The default value for the snapshot control file name is platform-specific, and dependent on the Oracle home. Manually reconfigure it to the correct value by using the RMAN command `CONFIGURE SNAPSHOT CONTROLFILE`.

Creating a Physical Standby Task 3: Create a Parameter File for the Standby Database

Create a parameter file (`PFILE`) from the server parameter file (`SPFILE`) used by the primary database.

To create a parameter file for the standby database, perform the following steps:

1. On the primary database, issue a SQL statement to create a copy of the primary database parameter file.

   In the following example,

   ```
   SQL> CREATE PFILE='/tmp/initboston.ora' FROM SPFILE;
   ```

2. Modify the parameter values in the copy parameter file as needed to use this copy as the parameter file for the standby database.

   Although most of the initialization parameter settings in the parameter file are also appropriate for the physical standby database, some modifications must be made.

Example 3-2 Modifying Initialization Parameters for a Physical Standby Database

This example shows the parameters created earlier on the primary that must be changed. The parameters that you must change are in bold typeface.

```
.
.
.
DB_NAME=chicago
```

```
DB_UNIQUE_NAME=boston
LOG_ARCHIVE_CONFIG='DG_CONFIG=(chicago,boston)'
CONTROL_FILES='/arch1/boston/control1.ctl', '/arch2/boston/control2.ctl'
DB_FILE_NAME_CONVERT='/chicago/','/boston/'
LOG_FILE_NAME_CONVERT='/chicago/','/boston/'
LOG_ARCHIVE_FORMAT=log%t_%s_%r.arc
LOG_ARCHIVE_DEST_1=
 'LOCATION=USE_DB_RECOVERY_FILE_DEST
  VALID_FOR=(ALL_LOGFILES,ALL_ROLES)
  DB_UNIQUE_NAME=boston'
LOG_ARCHIVE_DEST_2=
 'SERVICE=chicago ASYNC
  VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
  DB_UNIQUE_NAME=chicago'
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
STANDBY_FILE_MANAGEMENT=AUTO
FAL_SERVER=chicago
.
.
.
```

Ensure the `COMPATIBLE` initialization parameter is set to the same value on both the primary and standby databases. If the values differ, then redo transport services may be unable to transmit redo data from the primary database to the standby databases.

It is always a good practice to use the `SHOW PARAMETERS` command to verify that no other parameters need to be changed.

The following table provides a brief explanation about the parameter settings shown in that have different settings from the primary database.

Parameter	Recommended Setting
`DB_UNIQUE_NAME`	Specify a unique name for this database. This name uniquely identifies this database, and does not change even if the primary and standby databases reverse roles.
`CONTROL_FILES`	Specify the path name for the control files on the standby database. The example in this topic shows how to specify the path name for two control files. Oracle recommends that you ensure a copy of the control file is available, so that if a control file is corrupted, an instance can be easily restarted after copying the good control file to the location of the bad control file.
`DB_FILE_NAME_CONVERT`	Specify the path name and filename location of the primary database data files, followed by the standby location. The `CONTROL_FILES` parameter converts the path names of the primary database data files to the standby data file path names.
`LOG_FILE_NAME_CONVERT`	Specify the location of the primary database online redo log files followed by the standby location. This parameter converts the path names of the primary database log files to the path names on the standby database.

Parameter	Recommended Setting
LOG_ARCHIVE_DEST_*n*	Specify where the redo data is to be archived. In the example in this topic, the following destinations are specified: • LOG_ARCHIVE_DEST_1 archives redo data received from the primary database to archived redo log files in /arch1/boston/. • LOG_ARCHIVE_DEST_2 is currently ignored, because this destination is valid only for the primary role. If a switchover occurs, and this instance becomes the primary database, then this parameter specification provides the path to transmit redo data to the remote Chicago destination. **Note:** If a fast recovery area was configured (using the DB_RECOVERY_FILE_DEST initialization parameter), and you have not explicitly configured a local archiving destination with the LOCATION attribute, then Oracle Data Guard automatically uses the LOG_ARCHIVE_DEST_1 initialization parameter (if it has not already been set) as the default destination for local archiving.
FAL_SERVER	Specify the Oracle Net service name of the FAL (fetch archive log) server for a standby database. Typically, this service name is for the database running in the primary role. When the Boston database is running in the standby role, it uses the Chicago database as the FAL server from which to fetch (request) missing archived redo log files, if Chicago is unable to automatically send the missing log files.

> **Note:**
>
> Review the initialization parameter file for additional parameters that may need to be modified. For example, you may need to modify the dump destination parameters if the directory location on the standby database is different from those specified on the primary database.

Related Topics

- LOG_ARCHIVE_DEST_n Parameter Attributes

Upgrading Oracle Database with a Physical Standby Database in Place

These steps show how to upgrade to Oracle Database when a physical standby database is present in the configuration.

> **Note:**
>
> If the database being upgraded is a member of an Oracle Data Guard broker configuration, then before proceeding, you must disable fast-start failover and shut down the broker. For information about how to do this, see *Oracle Data Guard Broker* .

1. Review and perform the standard preupgrade preparation tasks described in *Oracle Database Upgrade Guide*.
2. Install the new release of the Oracle software into a new Oracle home on the physical standby database and primary database systems, as described in *Oracle Database Upgrade Guide*
3. Shut down the primary database.
4. Shut down physical standby databases.
5. Stop all listeners, agents, and other processes running in the Oracle homes that you want to upgrade (Source Oracle homes). Perform this step on all nodes in an Oracle Real Application Clusters (Oracle RAC) environment.
6. In the new Oracle home (Target Oracle home), restart all listeners, agents, and other processes that you stopped in the source Oracle home
7. Mount physical standby databases on the target Oracle home (upgraded version).

   > **Caution:**
   >
   > Do not open standby databases until the primary database upgrade is completed.

   See Start the Physical Standby Database for information on how to start a physical standby database.
8. Start Redo Apply on the physical standby databases.

   > **Note:**
   >
   > By default, AutoUpgrade disables log shipping. If you have modified your AutoUpgrade configuration file to enable log shipping, then modify your AutoUpgrade configuration file to set the AutoUpgrade locally modifiable global parameter `defer_standby_log_shipping` to `no`. For example: `upg1.defer_standby_log_shipping=no`

   See Start the Physical Standby Database for information on how to start Redo Apply.
9. Upgrade the primary database. Physical standby databases are upgraded when the redo generated by the primary database as it is upgraded is applied to standbys.
10. Open the upgraded primary database.
11. If Oracle Active Data Guard was being used before the upgrade, then you must reenable it after upgrading.

    See Real-time query
12. (Optional) When ready. modify the `COMPATIBLE` initialization parameter.

> **Note:**
>
> On Microsoft Windows platforms, it is necessary to use the ORADIM utility to delete the database service (for the old database version), and to create a new database service for the new database version. You must replace the `OracleService`*`SID`* on both the primary and standby servers.

Related Topics

- Oracle Data Guard Broker Upgrading and Downgrading
- Preparing to Upgrade Oracle Database, *Oracle Database Upgrade Guide*
- Modifying the COMPATIBLE Initialization Parameter After Upgrading

Creating a Physical Standby Task 4: Copy Files from the Primary System to the Standby System

Ensure that all required directories are created. Use an operating system copy utility to copy binary files from the primary system to their correct locations on the standby system.

Copy these binary files to the correct locations on the standby system:

1. The primary Oracle Database backup.

   See Create a Backup Copy of the Primary Database Data Files
2. the standby control file.

   See Create a Control File for the Standby Database
3. Standby database initialization parameter file.

   See Create a Parameter File for the Standby Database

Creating a Physical Standby Task 5: Set Up the Environment to Support the Standby Database

Set up the environment by creating a Windows-based service, a password file, and an `SPFILE`, and then setting up the Oracle Net environment.

To set up the environment, perform the following steps:

1. If the standby database is going to be hosted on a Windows system, then use the `ORADIM` utility to create a Windows service.

   For example:

   ```
   oradim -NEW -SID boston -STARTMODE manual
   ```

   The `ORADIM` utility automatically determines the username for which this service should be created and prompts for a password for that username (if that username needs a password).

   See *Oracle Database Administrator's Reference for Microsoft Windows* for more information about using the `ORADIM` utility.

2. Copy the remote login password file from the primary database system to the standby database system.

   This step is optional if operating system authentication is used for administrative users, and if SSL is used for redo transport authentication. If that is not the case, then copy the remote login password file from the primary database to the appropriate directory on the physical standby database system.

   Any subsequent changes to the password file on the primary are automatically propagated to the standby. Changes to a password file can include when administrative privileges (`SYSDG`, `SYSOPER`, `SYSDBA`, and so on) are granted or revoked, and when passwords of any user with administrative privileges is changed. Updated password files must still be manually copied to far sync instances because far sync instances receive redo, but do not apply it. Once the password file is up-to-date at the far sync instance, the redo containing the password update at the primary is automatically propagated to any standby databases that are set up to receive redo from that far sync instance. The password file is updated on the standby when the redo is applied.

3. Configure and start a listener on the standby system if one is not already configured.

   See Configuring and Administering Oracle Net Listener in *Oracle Database Net Services Administrator's Guide*.

4. Create Oracle Net service names.

   On both the primary and standby systems, use Oracle Net Manager to create a network service name for the primary and standby databases that are to be used by redo transport services. The Net service names in this example are `chicago` and `boston`.

   The Oracle Net service name must resolve to a connect descriptor that uses the same protocol, host address, port, and service that you specified when you configured the listeners for the primary and standby databases. The connect descriptor must also specify that a dedicated server be used.

   See Understanding Database Services in *Oracle Database Net Services Administrator's Guide* for more information about service names.

5. On an idle standby database, use the SQL `CREATE` statement to create a server parameter file for the standby database from the text initialization parameter file that was edited in Task 3.

   For example:

   ```
   SQL> CREATE SPFILE FROM PFILE='initboston.ora';
   ```

6. If the primary database has a database encryption wallet, then copy it to the standby database system and configure the standby database to use this wallet.

> **Note:**
>
> The database encryption wallet must be copied from the primary database system to each standby database system whenever the master encryption key is updated.
>
> Encrypted data in a standby database cannot be accessed unless the standby database is configured to point to a database encryption wallet or hardware security module that contains the current master encryption key from the primary database.

Creating a Physical Standby Task 6: Start the Physical Standby Database

These are the steps to start the physical standby database and Redo Apply.

1. On the standby database, issue the following SQL statement to start and mount the database:

   ```
   SQL> STARTUP MOUNT;
   ```

2. Restore the backup of the data files taken from the primary database data files, and copied to the standby system.

3. On the standby database, issue the following command to start Redo Apply:

   ```
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE -
   > DISCONNECT FROM SESSION;
   ```

   The statement includes the `DISCONNECT FROM SESSION` option, so that Redo Apply runs in a background session.

Creating a Physical Standby Task 7: Verify the Physical Standby Database Is Performing Properly

After you create the physical standby database and set up redo transport services, you may want to verify database modifications are being successfully transmitted from the primary database to the standby database.

To verify that redo is being transmitted from the primary database and applied to the standby database, connect to the standby database, and query the `V$DATAGUARD_PROCESS` view.

Example 3-3 Querying V$DATAGUARD_PROCESS to Verify Redo Transmission from Primary to Secondary Database

```
SQL> SELECT ROLE, THREAD#, SEQUENCE#, ACTION FROM V$DATAGUARD_PROCESS;


ROLE                        THREAD#  SEQUENCE# ACTION
------------------------ ---------- ---------- ------------
RFS ping                          1          9 IDLE
recovery apply applier            0          0 IDLE
recovery apply applier            0          0 IDLE
managed recovery                  0          0 IDLE
recovery logmerger                1          9 APPLYING_LOG
RFS archive                       0          0 IDLE
RFS async                         1          9 IDLE
```

The `recovery logmerger` role shows that redo is being applied at the standby.

> **Note:**
>
> Use the `V$DATAGUARD_PROCESS` view instead of the `V$MANAGED_STANDBY` view. `V$MANAGED_STANDBY` was deprecated in Oracle Database 12c Release 2 (12.2.0.1) and can be desupported in a future release.

4

Using AutoUpgrade for Oracle Database Upgrades

Learn how to use AutoUpgrade to simplify your upgrade tasks.

- About Oracle Database AutoUpgrade
  The AutoUpgrade utility is designed to automate the upgrade process, both before starting upgrades, during upgrade deployments, and during postupgrade checks and configuration migration
- 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.
- Understanding Unplug-Plug Upgrades with AutoUpgrade
  AutoUpgrade can perform an unplug of a pluggable database (PDB) from an earlier release source container database (CDB), plug it into a later release target CDB, and then complete all the steps required to upgrade the PDB to the target CDB release.
- AutoUpgrade Command-Line Parameters and Options
  Review the AutoUpgrade parameters and select the parameters and options for your Oracle Database upgrade use case.
- AutoUpgrade Utility Configuration Files Parameters and Options
  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 Patching
  AutoUpgrade has enhanced capability that enables it to automate all of the necessary steps in the database patching process, including optional downloading of patches.
- 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`.
- Known Restrictions for AutoUpgrade
  If you encounter issues with your upgrade, review the known restrictions to find solutions.
- 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.
- 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.
- 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 is designed to automate the upgrade process, both before starting upgrades, during upgrade deployments, and during postupgrade checks and configuration migration

When you perform upgrades, Oracle recommends that you download the most recent version of the AutoUpgrade utility (`autoupgrade.jar`) to prepare for and to deploy your upgrade. 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. You can now download the latest AutoUpgrade utility release directly from the Database Upgrades and Migrations page without having to log in to My Oracle Support: Download the AutoUpgrade utility

The `autoupgrade.jar` file exists by default in the Oracle home (*`Oracle_home`*`/rdbms/admin`). 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 the Database Upgrades and Migrations page.

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

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.

Try AutoUpgrade using our Hands-On Lab on Oracle LiveLabs

Oracle LiveLabs provides you with an environment in which you can try out Oracle technology in a free lab environment. Daniel Overby Hansen provides you with a demonstration of how you can use Oracle LiveLabs to set up your own demonstration lab environment in an Oracle Cloud environment. Check it out!

Try AutoUpgrade using our Hands-On Lab on Oracle LiveLabs

Related Topics

- Download the AutoUpgrade Utility (latest release)
- Database Upgrades and Migration
- AutoUpgrade Tool (Doc ID 2485457.1)
- Oracle LiveLabs

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.

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.

Because the source and target Oracle Database Oracle homes are on different servers, you run fixups on the source server, and the upgrade on the target 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.
- **SYSUPDATES**: The stage in which AutoUpgrade will bring up the Oracle RAC or an individual database for patching or upgrade.

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 either the source or target 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.

AutoUpgrade utility in Analyze mode runs `SELECT` statements in the database. It does not perform insert, update, or delete operations. However, be aware that if the database is opened in Read Write mode, then Oracle background processes can be writing to the database to support necessary database functionality. 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.

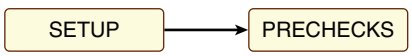
You start AutoUpgrade in Analyze mode using the following syntax, where *`Java-8-home`* is the location of your Java 8 distribution, or the environment variable set for the Java 8 home, and *`path/yourconfig.txt`* is the path and filename of your configuration file:

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

For example, suppose you have copied the most recent AutoUpgrade release to the new release Oracle home under rdbms/admin, and set an environment variable for that home to `21CHOME`, and copied the configuration file under the Oracle user home, under the directory `/scripts`, and called it `21config.cfg`, you then enter the following command:

```
java -jar $21CHOME/rdbms/admin/autoupgrade.jar -config /scratch/scripts/
21config.cfg -mode analyze -mode analyze
```

Oracle Database Release 12.2 (12.2.0.1) or newer Oracle homes have a valid java version by default.



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\12.2.0\dbhome_1
#
# Target Oracle Home
#
sales3.target_home=d:\app\oracle\product\21.0.0\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/jdk/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/jdk/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 either the source or target Oracle home.

You can use the Upgrade mode to divide an upgrade into two parts:

1. (Strongly Recommended) Run the Prefixups mode on the source database running on its Oracle home.
2. (Optional) Move the source database to a new Oracle home on a different system.
3. Perform the upgrade of the database using the Upgrade mode.

> **Note:**
>
> When run in the source Oracle home, AutoUpgrade will start processing the upgrade immediately after skipping the PRECHECKS and PREFIXUPS stages. All other stages (except `POSTUPGRADE`) typically run during a `DEPLOY` will be run.

To use the Upgrade mode, the database must be up and running in either the source or the target Oracle home before you run AutoUpgrade in Upgrade mode. This option is particularly of value 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.

This procedure runs the upgrade, and postfixups operations on the database in the new Oracle home location.

Upgrading PDBs with Upgrade Mode

To upgrade a PDB, the folllowing requirements must be met:

- The PDB must already be added to the CDB.
- The PDB must already be opened in Upgrade mode.
- If you ran a manual Non-CDB to PDB or Unplug-Plug procedure, these processes must be fully completed before you run AutoUpgrade.
- When the PDB was not previously added to the CDB, and that PDB had a Transparent Data Encryption (TDE) configuration, you must reimport the TDE after the upgrade is completed.

AutoUpgrade Upgrade Mode in Target Oracle Home

In a CDB, when `CDB$ROOT` is in a major version (first numeral), we can use Autoupgrade to upgrade the PDBs that are in a lower version. The process for each PDB that you upgrade is as follows:

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.

> **Note:**
>
> When you run AutoUpgrade in Upgrade mode, PDBs must already be open in migration mode. There is no automated backup option for this configuration. In this scenario, postupgrade operations are not performed, so you must complete those steps separately later. For example, the following postupgrade operations are not performed:
>
> - Copy of network files (`tnsnames.ora`, `sqlnet.ora`, `listener.ora` and other listener files, LDAP files, `oranfstab`
> - Removal of the guaranteed restore point (GRP) created during the upgrade
> - Final restart of an Oracle Real Application Clusters database

AutoUpgrade Upgrade Mode in Source Oracle Home

When the database in open in the source Oracle home, the stages run in the upgrade home depend on whether Fixups have been run on the source Oracle home before you start AutoUpgrade in Upgrade mode:

- If Fixups have already been run on the Source Oracle home, then all of the stages of a typical Deploy mode are run, except for Prechecks and Prefixups.
  Use this option if you can run the Prechecks and Prefixups separately, because AutoUpgrade bypasses running the Prechecks and Prefixups stages during the upgrade itself, which reduces your downtime.
- If fixups on the source Oracle home have not been run within the previous 3 days, then Upgrade mode includes those stages. The result is that running AutoUpgrade in Upgrade mode on the source Oracle home is exactly the same as running AutoUpgrade in Deploy mode, because the Prechecks and Prefixups stages are run as part of the Upgrade mode.

Example 4-1 Running AutoUpgrade in the Target Home After Moving the Database to a New Location

Where *dbname* is the name of your database, you run AutoUpgrade using the following steps:

1. - If you ran AutoUpgrade with the Prefixups mode:
     a. Copy the `during_upgrade_pfile_`*dbname*`.ora` file to the default location in the target Oracle home with the default name (`init`*SID*`.ora`).

        The `during_upgrade_pfile_`*dbname*`.ora` file is located under the `temp` directory in the log path used to run AutoUpgrade.

     b. (Optional) You can connect to SQL*Plus and create an SPFILE using `during_upgrade_pfile_`*dbname*`.ora` in the `temp` directory. For example:

        ```
        SQL> create spfile from pfile='/u01/autoupgrade/au21/CDBUP/temp/
        during_upgrade_pfile_cdbupg.ora';
        ```

   - If you did not run AutoUpgrade with the Prefixups mode:
     a. Copy the initialization file (`init.ora` or `spfile`*SID*`.ora` from the source Oracle home to the target Oracle home location.

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

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

   This command runs the upgrade operations on the database.

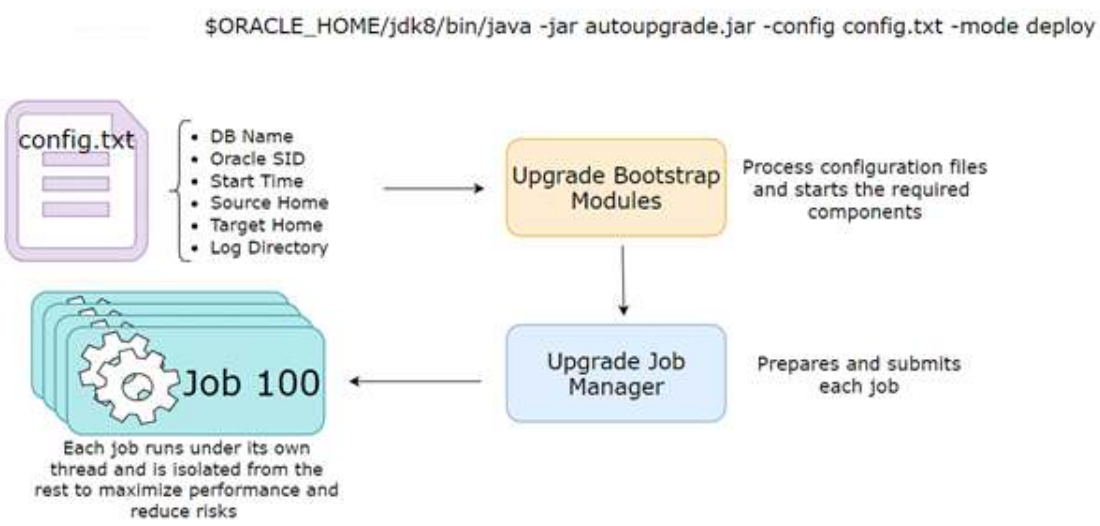
Understanding AutoUpgrade Workflows and Stages

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.

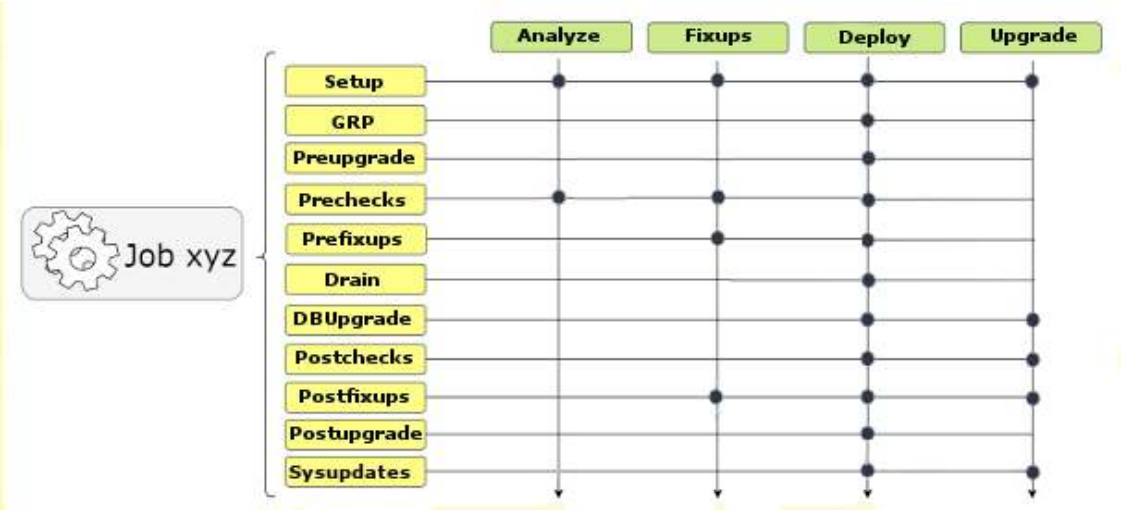
AutoUpgrade Processing Mode Workflow Processing

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, Postfixups, Postupgrade, and Sysupdates. 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, Postfixups, and Sysupdates.

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.

Starting with Oracle Database 21c, all upgrades must use the multitenant architecture. Use of the non-CDB Oracle Database architecture is desupported. When you migrate your database from the non-CDB architecture to PDBs, 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.

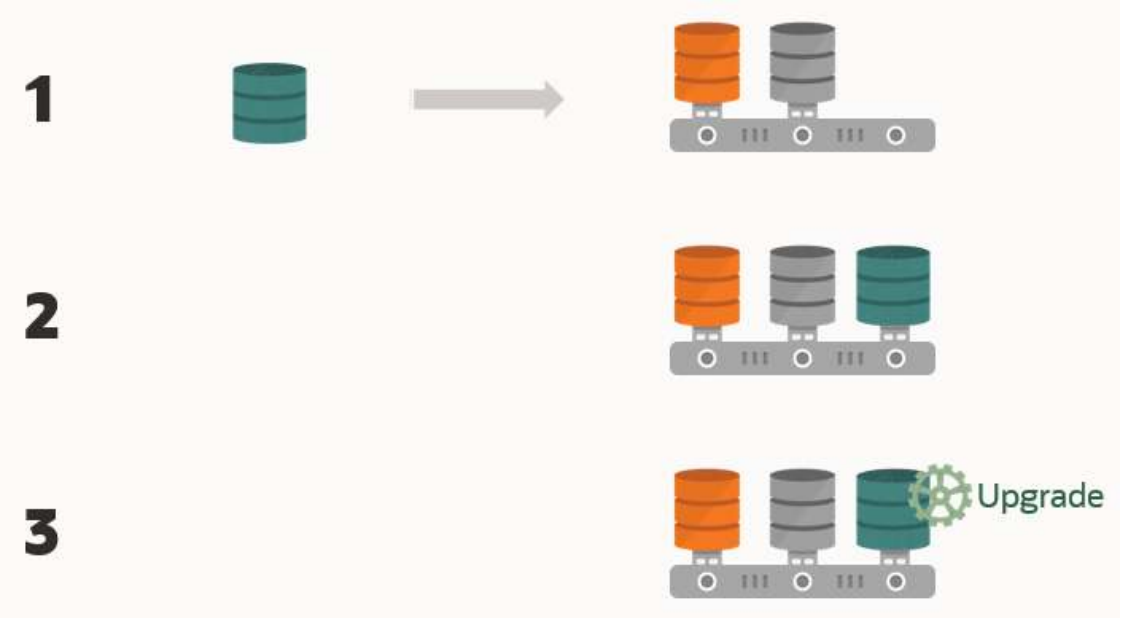
The non-CDB to PDB feature of the AutoUpgrade utility provides you flexible options to control how you upgrade your earlier release non-CDB Oracle Database when you upgrade and convert to the multitenant architecture. Starting with Oracle Database 21c, when you have an existing target release CDB, you can use AutoUpgrade to convert a non-CDB Oracle Database to a PDB on the target release CDB during the upgrade. To perform an upgrade and conversion of the non-CDB to a PDB, you provide information about your non-CDB in the AutoUpgrade configuration file. If you prefer, you can also choose to convert your non-CDB Oracle Database to a PDB in the source release, and then plug in the PDB to a target release CDB, where the upgrade is performed when you plug in the PDB.

> **Caution:**
>
> Before you run AutoUpgrade to complete the conversion and upgrade. Oracle strongly recommends that you create a full backup of your source database, and complete thorough testing of the upgrade. There is no option to roll back to the non-CDB Oracle Database state after AutoUpgrade starts this procedure.

Figure 4-1 Converting a Non-CDB to a PDB and Upgrading the PDB Using AutoUpgrade

In the following illustration, a non-CDB Oracle Database goes through the following steps:

1. AutoUpgrade uses the information you provide in the configuration file to move the non-CDB source release database to the target release Oracle Database.
2. The source database is converted to a PDB on the target release.
3. The source database (now a PDB) is upgraded to the target release.



Requirements for Source Non-CDB and Target CDB

Requirements on the source non-CDB and target CDB to perform upgrades and conversions to PDBs are as follows:

- The target CDB must be created in advance of performing the upgrade with AutoUpgrade.
- The PDB created from the non-CDB must continue to use the source non-CDB name. You cannot change the name of the database.
- The set of Oracle Database options in the target database must be either be the same as in the source database, or a superset of the options in the source database.
- The endian format of the source non-CDB and target CDB are identical.
- The source non-CDB and target CDB have compatible character sets and national character sets.

- The source non-CDB Oracle Database release and operating system platform must be supported for direct upgrade to the target CDB release.
- Operating system authentication is enabled for the source non-CDB and target CDB.

The minimum `COMPATIBLE` parameter setting for the source database must be at least 12.2.0. If the `COMPATIBLE` setting is a lower version, then during the conversion and upgrade process, `COMPATIBLE` is set to 12.2.0. During the conversion, the original datafiles are retained. They are not copied to create the new PDB. To enable AutoUpgrade to perform the upgrade, edit the AutoUpgrade configuration file to set the AutoUpgrade parameters `target_version` to the target CDB release, and identify the CDB to which the upgraded database is placed using `target_cdb`. During the conversion and upgrade process, AutoUpgrade uses that information to complete the upgrade to the target CDB.

Example 4-2 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/21.1.0/dbhome_1
upg1.target_cdb=cdb21x
```

You can see a more detailed example of a non-CDB to PDB upgrade from Oracle Database 12c (12.2) to Oracle Database 19c using the multitenant architecture in the blog post "Unplug / Plug / Upgrade with AutoUpgrade," in Mike Dietrich's Blog, *Upgrade Your Database Now!*, and also a demonstration of the noncdb_to_pdb.sql automatic feature using AutoUpgrade, in *AutoUpgrade to Oracle Database 19c - and plug into a CDB*.

Related Topics

- Unplug / Plug / Upgrade with AutoUpgrade in Mike Dietrich, Upgrade Your Database Now
- AutoUpgrade to Oracle Database 19c - and plug into a CDB
- Permitted Features, Options, and Management Packs by Oracle Database Offering

Understanding Unplug-Plug Upgrades with AutoUpgrade

AutoUpgrade can perform an unplug of a pluggable database (PDB) from an earlier release source container database (CDB), plug it into a later release target CDB, and then complete all the steps required to upgrade the PDB to the target CDB release.

There are two workflows for unplug-plug PDB upgrades using AutoUpgrade, depending on how you configure the upgrade:

- You unplug one or more pluggable databases from one source CDB, and plug them into a new release target CDB
- You unplug multiple pluggable databases from different source CDBs, and plug them into a new release target CDB

In addition, for unplug-plug operations, AutoUpgrade now supports moving the default state of a PDB from the source PDB to the target PDB. If you set `alter pluggable database save`

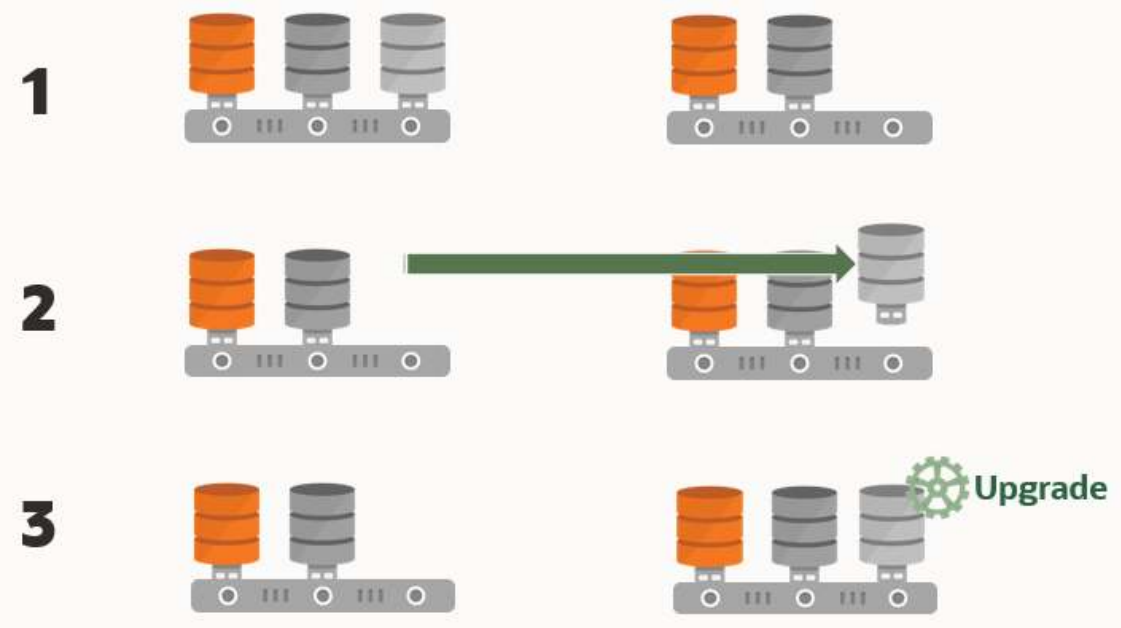
`state` on a source PDB, then that state is transferred to the target PDB, so that the PDB is automatically opened when CDB$ROOT is opened. You can also control whether AutoUpgrade keeps or discards the database state by using the AutoUpgrade configuration file local parameter `keep_pdb_save_state`.

> **Caution:**
>
> As with any other change to the database, before you run AutoUpgrade to complete the conversion and upgrade, Oracle strongly recommends that you implement a reliable backup strategy to prevent unexpected data loss. There is no option to roll back an unplug-plug PDB upgrade after AutoUpgrade starts this procedure. Flashback Database also does not work across the PDB conversion, and is not reversible. Backups are the only fallback strategy.

The following illustration shows the unplug-plug operation, in this case of a single PDB:

1. There is one source Oracle Database, and one target release Oracle Database. At this stage, create your configuration file and run AutoUpgrade in Analyze mode (`autoupgrade.jar -mode analyze`) to check your readiness for upgrade, and to correct any issues that are reported.
2. You run AutoUpgrade in Deploy mode (autoupgrade.jar -mode deploy). AutoUpgrade uses the information you provide in the configuration file to move the PDB to the target release, and plug in the PDB.
3. AutoUpgrade runs prefixups, and then upgrades the PDB to the target release.

Figure 4-2 Unplug-Plug Upgrades from Source to Target



Requirements for Source and Target CDBs

To perform an unplug-plug upgrade, your source and target CDBs must meet the following conditions:

- You have created the target release CDB, and opened the CDB before starting the unplug-plug upgrade.
- The endian format of the source and target CDBs are identical.
- The set of Oracle Database components configured for the target release CDB include all of the components available on the source CDB.
- The source and target CDBs have compatible character sets and national character sets
- The source CDB release must be supported for direct upgrade to the target CDB release.
- External authentication (operating system authentication) is enabled for the source and target CDBs
- The Oracle APEX installation type on the source CDBs should match the installation type on the target CDB.
- There should be no existing guaranteed restore point (GRP) on the non-CDB Oracle Database that you want to plug in to the CDB.

> **Note:**
>
> With AutoUpgrade 22 and later updates, you can now use AutoUpgrade to plug into an Oracle Data Guard configuration. AutoUpgrade creates the PDB with the `STANDBYS=NONE` clause. After the upgrade, you can re-establish standbys by recovering the data files on the standby databases.

Features of Unplug-Plug Upgrades

When you select an unplug-plug upgrade, depending on how you configure the AutoUpgrade configuration file, you can use AutoUpgrade to perform the following options during the upgrade:

- You can either keep the PDB name that you have in the source CDB, or you can change the PDB name.
- You can make a copy of the data files to the target CDB, while preserving all of the old files.
- You can copy the data files to the target location, and then delete the old files on the source CDB
- You can process one PDB, or you can link to an inclusion list and process many PDBs in one upgrade procedure; the only limit for the number of PDBs you can process are the server limits, and the limits for PDBS on the CDB.

Example 4-3 AutoUpgrade Configuration File for Unplug-Plug Upgrades

To use the unplug-plug PDB upgrade option, you must identify the following values in the AutoUpgrade configuration file:

- The system identifier parameters for the source CDB (parameter `sid`).
- The target CDB (parameter `target_cdb`).
- The name of the PDB in the source CDB, and, if you want to convert it, the target conversion name.

For example, where the source CDB is `CDB122`, the target CDB is `cdb21x`, the name of the PDB in the source CDB is `pdb2`, and the conversion name for the PDB that you want on the target CDB is `depsales`:

```
global.autoupg_log_dir=/home/oracle/autoupg
upg1.sid=CDB122
upg1.source_home=/u01/app/oracle/product/12.2.0/dbhome_1
upg1.target_home=/u01/app/oracle/product/19.1.0/dbhome_1
upg1.target_cdb=cdb21x
upg1.pdbs=pdb_2
upg1.target_pdb_name.pdb_2=depsales
upg1.target_pdb_copy_option.pdb_2=file_name_convert=('pdb_2','depsales')
```

AutoUpgrade Command-Line Parameters and Options

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 checkpoint, which causes AutoUpgrade to have a fresh start the next time the tool is launched 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.
- listchecks
- load_password
- load_win_credential

- 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.
- Patch
  The AutoUpgrade parameter `patch` is used to specify that AutoUpgrade is performing a patch operation, and not an upgrade operation.
- 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

- AutoUpgrade is compatible with JDK 8 to JDK 11. You must have at least Java Development Kit (JDK) 8 or a later JDK release 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.

  Starting with Oracle Database 23ai, every Oracle home includes Java Runtime Environment (JRE) 11. AutoUpgrade is compiled with Java 11, and is compatible with JDK 8 to JDK 11.
- 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
`-restore_on_fail`	If set, then when a job fails, the database is restored automatically. Errors in PDBs are not considered irrecoverable, only errors in `CDB$ROOT` or Non-CDBs.

Parameter	Description
`-console`	Starts AutoUpgrade with the console enabled.
`-noconsole`	Starts AutoUpgrade with the console disabled.
`-proceed`	Alters the predefined `start_time` on scheduled jobs, or jobs during the `REFRESHPDB` stage For a full description of this parameter, see the `-proceed` parameter topic.
`-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.
`restore_on_fail`	automatically restores any job that failed during the deployment.
`-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.
`-listchecks`	Provides a list of all upgrade checks for an upgrade, or if you specify a particular check, the details about the check you specify.
`-load_password`	Enables you to enter passwords AutoUpgrade requires safely into AutoUpgrade's keystore.
`-load_win_credential`	Uses PowerShell to create a credential object so that AutoUpgrade can be run without interruption during the upgrade.
`-patch`	Specifies that AutoUpgrade is performing a patch operation, and not an upgrade operation.
`-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.
-settings	Identifies the configuration file that you use to provide custom runtime configuration of the AutoUpgrade utility.

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 checkpoint, which causes AutoUpgrade to have a fresh start the next time the tool is launched 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 three arguments:

- The configuration file name
- (Optional) The path to the configuration file, as represented by *config-file*

  When used in conjunction with the parameter `-load_password`, AutoUpgrade also creates a keystore for Transparent Data Encryption (TDE) passwords in the location specified by the global configuration file parameter `global.keystore`, if that parameter is set in the configuration file for a database.

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/19.0.0/`
  - Microsoft Windows: Equivalent to a `SET ORACLE_TARGET_HOME` command. For example: `SET ORACLE_TARGET_HOME=C:\oracle\19`
- `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/19.0.0/dbhome_1`
- `ORACLE_SID` is set to `sales`
- `ORACLE_TARGET_VERSION` is set to `19.1`

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/19
# Value from environmental variable ORACLE_SID
autoupgrade1.sid=sales
# Value from environmental variable ORACLE_TARGET_VERSION
autoupgrade1.target_version=19.3
```

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/19
autoupgrade1.target_home=/scratch/21
autoupgrade1.sid=sales
autoupgrade2.source_home=/scratch/19
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 enable or disable the AutoUpgrade console using the options `-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.

You can use the proceed option to have AutoUpgrade start a job at a time that you specify. You have three options:

- Specify no start time (default: Delay start by one minute after the command is run)
- Specify a start time delay with an hour and minute value from the time the command is run
- Specify a start delay with a day, month, year, minute, and second delay value from the time the command is run

> **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)-a number] -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. `-a number`: (Optional) Repeats the command after the number of seconds specified by integer value (*number*). `-n number`: (Required) Number of jobs to display, specified by integer value.
`-lsr`	Displays the restoration queue.
`-lsa`	Displays the queue of jobs to stop.

Console Option	Description
-proceed -job *job_number* -newStartTime *values*	Enables you to alter the predefined start_time on scheduled jobs or jobs during the REFRESHPDB stage. Use this feature with the AutoUpgrade hot cloning and relocate feature. You must specify a job number by using the -job_number flag. The -proceed parameter can only be used with the AutoUpgrade Console. You cannot use -proceed in -noconsole mode. You can enter -newStartTime to specify a specific time. If no value for -newStartTime is specified, then by default the start time is delayed by one minute from the time the command is run for the job specified. • *job_number* specifies the job number where you want to alter the start time. • When specifying a different start time using the hour and minute, the following arguments apply: #h specifies the numeric value of how many hours the job should be delayed. For example: 10h specifies a ten hour delay from the time the command is run. #m is the numeric value of how many minutes past the hour you specify that the job should be delayed. For example, when the specification is +10h17m, the start time is specified as a ten hours and seventeen minutes delay from the time the command is run. • When specifying a different date and time start, specify the delay using *dd/mm/yyyy hh:mm:ss* format, where the delay is by day, month, and year, and the time in 24-hour format by hour, minutes and seconds upg> proceed -job 100 -newStartTime +10h17m
-tasks	Displays the tasks that are running.
-clear	Clears the terminal display.
-resume -job *number* [-ignore_errors=ORA-*number*,ORA-*number*]	Restarts from a previous job that was running, specified by a numeric value (*number*) for the job. ignore_errors option: This flag is optional. If there are any resume errors during patching or upgrade stage processing, then that is reported as a stage failure. AutoUpgrade will not proceed to postupgrade operations. There are some cases where an error can be ignored, and postupgrade operations can continue. If you think that a specific error will not affect patching or upgrade, then you can use the ignore_errors option to specify the errors that you want to ignore, so that postupgrade operations can continue. The error numbers are specified by a comma-delimited list of errors. Example: -resume -job 444 -ignore_errors=ORA-48101,ORA-00001

Console Option	Description
`-status [-job number | -c dbname| -config| -a number]`	Lists the status of a particular job with the response you specify with the flag. Flags: `-job number`: Shows information about a specific job, specified by a numeric value. `-c dbname`: Displays information about the specific database name that you specify (`dbname`), with detailed information, if available. `-config` *number* Displays configuration information for the job that you specify. `-a number`: (Optional) Repeats the command after the number of seconds specified by integer value (`number`).
`-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`	Stops 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] [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`. You can add a clause to specify the type of AutoUpgrade configuration file, using 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
```

listchecks

The AutoUpgrade parameter `listchecks` either provides a list of all upgrade checks for an upgrade, or if you specify a particular check, the details about the check you specify.

Property	Description
Parameter type	string
Syntax	`-listchecks [checkname]`
Default value	None. If no specific check is specified, then a list of all AutoUpgrade checks is provided.

Examples

When entered without a specification, the Autoupgrade `listchecks` parameter generates descriptions of all of the checks AutoUpgrade performs for the upgrade.

```
$ java -jar autoupgrade.jar  -listchecks
Check : AMD_EXISTS
        Description : Starting with Oracle Database 12c, the OLAP Catalog
(OLAP AMD) is desupported and will be automatically marked as OPTION OFF
during the database upgra
de if present. Oracle recommends removing OLAP Catalog (OLAP AMD) before
database upgrade.  This step can be manually performed before the upgrade to
reduce downtime.
        Fixup Action : Remove OLAP Catalog by running the {1} SQL
script $ORACLE_HOME/olap/admin/catnoamd.sql script.
        Severity : WARNING
```

```
        Fixup Stage : PRE
        Min Version(inclusive) Check applies : NONE
        Max Version(exclusive) Check applies : NONE
        Check Introduced Version : NONE
        Check Removed Version : NONE
        Manual Fixup or Automatic : AUTO
        AutoUpgrade Only : NO
        Run for Datapatch : NO


Check : APEX_MANUAL_UPGRADE
        Description : Starting with Oracle Database Release 18, APEX is not
upgraded automatically as part of the database upgrade. Refer to My Oracle
Support Note 1088970.
1 for information about APEX installation and upgrades. Refer to MOS Note
1344948.1 for the minimum APEX version supported for your target database
release. Unsupported ver
sions of APEX will be in an INVALID state when its database dependencies are
not in sync with the upgraded database.
        Fixup Action : Upgrade Oracle Application Express (APEX) manually
before or after the database upgrade.
        Severity : WARNING
        Fixup Stage : PRE
        Min Version(inclusive) Check applies : NONE
        Max Version(exclusive) Check applies : NONE
        Check Introduced Version : 18
        Check Removed Version : NONE
        Manual Fixup or Automatic : MANUAL
        AutoUpgrade Only : NO
        Run for Datapatch : NO


Check : APEX_PATCH
        Description : The APEX patching process is not performed by the {1}
Oracle database upgrade.  The APEX version upgrade only ensures that the APEX
version is upgrade
d to version {3} and does not guarantee the version is brought all the way to
the patched level {2}.  If a PDB from this CDB is unplugged and plugged into
another ROOT, the
.
.
.
```

When entered with a specified check, `listchecks` provides details about the checks for the check you specify:

```
$ java -jar autoupgrade.jar  -listchecks XDB_RESOURCE_TYPE
Check : XDB_RESOURCE_TYPE
        Description : Direct access to either TYPE XDB.XDB$RESOURCE_T or
TABLE XDB.XDB$RESOURCE is restricted to Oracle internal code only.
        Fixup Action : Please contact Oracle Support to resolve the problem.
        Severity : ERROR
        Fixup Stage : PRE
        Min Version(inclusive) Check applies : 11.1
        Max Version(exclusive) Check applies : NONE
```

```
Check Introduced Version : NONE
Check Removed Version : NONE
Manual Fixup or Automatic : MANUAL
AutoUpgrade Only : NO
Run for Datapatch : NO
```

load_password

The AutoUpgrade parameter `load_password` enables you to enter passwords safely into AutoUpgrade's keystore. When you run AutoUpgrade in analyze mode, you are notified which passwords are needed, and can be loaded into the AutoUpgrade keystore.

Property	Description
Parameter type	string
Syntax	`-load_password`
Default value	None. AutoUpgrade prompts you for password input values in an interactive prompt.

Description

To provide passwords required for upgrade, you can use the `load_password` parameter. This parameter must be used in conjunction with the `-config` parameter and the configuration file global parameter `keystore`, to designate the location of the dedicated software keystore where AutoUpgrade securely stores passwords. The `-load_password` parameter takes no arguments. Instead, it starts an interactive prompt with specific commands that enable you to provide information required for the keystore. This option can be used to add Transparent Data Encryption passwords (TDE), My Oracle Support credentials (MOS), Oracle Recovery Manager (RMAN) credentials, and database SYS passwords (PWD). The command options are slightly different for each of these use cases. Common options include the following:

- `group [tde|pwd|rman|mos]`
  Specifies the scope of the `load_password` option:
  - `mos`: Specifies that the `load_password` options in the group apply to the My Oracle Support user (MOS). This group option applies only when AutoUpgrade is used with the `-patch` command line option.
  - `pwd`: Specifies that the `load_password` options in the group apply to the specified database SYS password (PWD). This group option applies to both upgrades and patching.
  - `tde`: Specifies that the `load_password` options in the group apply to Transparent Data Encryption (TDE). This option applies for both upgrades and patching.
  - `rman`: Specifies that the `load_password` options in the group apply to the specified database RMAN password (RMAN). This group option applies only when AutoUpgrade is not used with the `-patch` command line option.
- `help`

  Lists all the available `load_password` commands.
- `exit`

Exits the `load_password` interactive console. If the keystore has been modified and not yet saved, then you are prompted to determine if you want to save the keystore before exiting.

During the upgrade, AutoUpgrade places passwords in an encrypted password array in memory, so that AutoUpgrade can access source database keystores and resources. No passwords are written to SQL*Plus scripts during the upgrade. After AutoUpgrade no longer requires the passwords, these passwords are purged from memory. No log records are kept of the passwords.

Transparent Data Encryption (TDE) Group Options

> **Note:**
>
> - If you configure an Keystore External Password Store in the database, then AutoUpgrade detects the presence of an Keystore External Password Store, and uses this external password store instead of requiring manual input of the TDE keystore passwords. However, if all databases are configured with Keystore External Password Store, then in some situations, you can still need to define global.keystore.
> - In some situations, AutoUpgrade needs access to the AutoUpgrade keystore to write other sensitive information. For example, AutoUpgrade can write transport secrets (passphrases) that are used by `ADMINISTER KEY MANAGEMENT EXPORT KEYS` and `ADMINISTRATER KEY MANAGEMENT IMPORT KEYS` commands to AutoUpgrade.
>
> For more information about Keystore External Password Stores, refer to *Oracle Database Advanced Security Guide* or *Oracle Database Data Redaction Guide*.

When you run AutoUpgrade using `-mode analyze`, AutoUpgrade detect which passwords are needed for the databases specified for upgrade in your configuration file, and lists them in the preupgrade summary report. Before the upgrade, you can then use `-load_password` to enter the passwords for the databases. These passwords are stored safely in AutoUpgrade's own keystore, in the location specified by `global.keystore`. The passwords are used only to access the source database TDE keystores, and to write the TDE passwords in the target keystores.

> **Caution:**
>
> Because the directory you specify AutoUpgrade to create with `global.keystore` contains a software keystore, it should be protected using the same security best practices as you use with TDE keystore files.

When you run AutoUpgrade with the `-load_parameter` option at the command line, AutoUpgrade starts an interactive console so that you can configure password options. The console indicates if you need

- `add` *`ORACLE_SID`* `[-pdb` *`isolated-pdb`*`]`

  Adds a TDE password for the specified Oracle System identifier (*`ORACLE_SID`*).

  If you have an isolated PDB that requires a password, then use the optional `-pdb` parameter in the configuration file to specify an isolated PDB for which you want to provide

a password. If the CDB root and all PDBs are configured in united mode, then the `-pdb` parameter is not required, as the keystore is shared between the CDB root and all PDBs that are configured in united mode.

- `delete ORACLE_SID [-pdb isolated-pdb]`

  Deletes a loaded password for the specified Oracle System identifier (*`ORACLE_SID`*).

  If you loaded a password for an isolated PDB that you want to delete, then use the optional `-pdb pdbname` parameter in the configuration file to specify the name of the isolated PDB whose password you want to delete. If the CDB root and all PDBs are configured in united mode, then the `-pdb` parameter is not required.

- `list`

  Lists each Oracle Database by Oracle System Identifier (*`ORACLE_SID`*), provides details for each database, and indicates if further actions are necessary to perform an encrypted database upgrade. AutoUpgrade starts a deploy mode only when there are no pending actions for any of the databases listed in the configuration file. If an action is required before a database can be upgraded, then the AutoUpgrade check `tde_passwords_required` fails during the prechecks stage.

My Oracle Support (MOS) Credential Group Options

- `add -user email-address`

  The `add -user email-address` option stores My Oracle Support credentials in the AutoUpgrade Patching keystore, where *`email-address`* is the user email you specify. When `download=YES`, the credentials are used to connect to My Oracle Support so that AutoUpgrade can download the required patches.

- `list` displays whether My Oracle Support credentials have been loaded into the keystore, and if so, whether the loaded credentials could be used successfully to connect to My Oracle Support. You can only load one set of credentials.

- `delete -user` deletes the My Oracle Support credentials.

Oracle Recovery Manager (RMAN) Credential Group Options

- `add ORACLE_SID -user username`

  Adds the username and password for the specified Oracle System identifier (*`ORACLE_SID`*).

- `delete ORACLE_SID -user username`

  Deletes username and password for the specified Oracle System identifier (`ORACLE_SID`).

- `list`

  Lists each Oracle Database by Oracle System Identifier (*`ORACLE_SID`*), provides details for each database, and indicates if further actions are necessary.

- `pwd`

  Specifies the RMAN password needed to connect to the specified database.

Database SYS Password (PWD) Credential Group Options

- `add ORACLE_SID -user username`

  Adds the username and password for the specified Oracle System identifier (*`ORACLE_SID`*).

- `delete ORACLE_SID -user username`

  deletes username and password for the specified Oracle System identifier (`ORACLE_SID`).

- `list`

  Lists each Oracle Database by Oracle System Identifier (*`ORACLE_SID`*), provides details for each database, and indicates if further actions are necessary to perform an encrypted database upgrade. AutoUpgrade starts a deploy mode only when there are no pending actions for any of the databases listed in the configuration file. If an action is required before a database can be upgraded, then the AutoUpgrade check `tde_passwords_required` fails during the prechecks stage.

- `pwd`

  Specifies the SYS password needed to connect to the specified database. Sampe options as RMAN

- `exit`

  Exits the `load_password` interactive console. If the keystore has been modified and not yet saved, then you are prompted to determine if you want to save the keystore before exiting.

Examples

TDE Keystore Passwords Added to AutoUpgrade Keystore

Run AutoUpgrade to add the TDE keystore passwords to AutoUpgrade's own keystore, using a configuration file named `myconfig.cfg`, where `-load_password` is used to prompt you for TDE passwords of any database where the TDE keystore password is needed:

```
java -jar autoupgrade.jar -config myconfig.cfg -load_password
```

After the TDE passwords are loaded, you can then either run AutoUpgrade in `analyze` or config `mode`:

```
java -jar autoupgrade.jar -config myfile.cfg -mode deploy
```

AutoUpgrade uses the TDE password files from its own keystore to access the source database TDE keystores, and to write the TDE passwords in the target database keystores.

Multiple Load Password Options Used with Multiple Source Database Upgrades

In the following example, all of the `load_password` command options are used to load TDE passwords from source databases `db12201`, `cdb122`, and `db19x` to the common target CDB `cdb19x`:

```
$ java -jar autoupgrade.jar -config config.cfg -load_password
Processing config file ...

Starting AutoUpgrade Password Loader - Type help for available options
Creating new keystore - Password required
Enter password:
Enter password again:
Keystore was successfully created

TDE> add cdb19x
Enter your secret/Password:
Re-enter your secret/Password:
TDE> add cdb122
Enter your secret/Password:
```

```
Re-enter your secret/Password:
TDE> add db12201
Enter your secret/Password:
Re-enter your secret/Password:
TDE> add db19x
Enter your secret/Password:
Re-enter your secret/Password:
TDE> delete cdb19x
TDE> list
+----------+---------------+-----------------+----------+-----------------
+
|ORACLE_SID|Action Required | TDE Password     |SEPS Status|Active Wallet
Type|
+----------+---------------+-----------------+----------+-----------------
+
|    cdb122|               |          Verified|   Inactive|
Any|
|    cdb19x|Add TDE password|No password loaded|   Inactive|
Any|
|   db12201|               |          Verified|   Inactive|        Auto-
login|
|     db19x|               |          Verified|   Inactive|
Any|
+----------+---------------+-----------------+----------+-----------------
+
TDE> help
The following options are available
-----------------------------------
1 add
2 delete
3 list
4 group
5 save
6 help
7 exit

TDE> save
Convert the keystore to auto-login [YES|NO] ? YES
TDE> exit

AutoUpgrade Password Loader finished - Exiting AutoUpgrade
```

Adding an isolated PDB TDE Keystore Password Added to AutoUpgrade Keystore

In isolated mode, where a pluggable database (PDB) has its own keystore, PDBs are allowed to independently create and manage their own keystore. For isolated mode PDBs, start the AutoUpgrade Password Loader, and use the syntax `add` *`Oracle_SID`* `-pdb` *`pdbname`*, where *`Oracle_SID`* is the name of the CDB root, and *`pdbname`* is the name of the isolated PDB.

For example, where CDB root is `cdb19x` and the isolated PDB name is `iso`:

```
TDE> add cdb19x -pdb iso
Enter your secret/Password:
Re-enter your secret/Password:
```

Related Topics

- About Configuring United Mode
- About Configuring an External Keystore
- Configuring Auto-Open Connections into External Key Managers

load_win_credential

The AutoUpgrade parameter `load_win_credential` uses PowerShell to create a credential object so that AutoUpgrade can be run without interruption during the upgrade.

Description

To provide passwords required for upgrade on Microsoft Windows platforms, you can use the `load_win_credential` parameter. This parameter must be used in conjunction with the `-config` parameter, and with a configuration file that specifies the local `wincredential` parameter. The parameter starts up a Microsoft Windows PowerShell credential request, where you can provide Administrator credentials to create a PowerShell credential object. By default, the Windows PowerShell Credential request is for the local machine. You provide the user name for the credential (the owner of the Oracle binaries) and that user password. Windows Powershell then generates a credential using the format database-name.xml, where database-name is the name of the database that you specified in your local configuration file SID entry. For example, `upg1.sid=sales1` in the configuration file with the `wincredential` entry `upg1.wincredential=C:\Users\oracle\cred` generates the powershell credential file `sales1.xml` in the path `sales1.wincredential=C:\Users\oracle\cred`

After AutoUpgrade no longer requires the passwords, these passwords are purged from memory. No log records are kept of the passwords.

> **Note:**
>
> If you do not provide a system identifier (SID) as an option to the `-load_win_credential` parameter, and if only one database is specified in the configuration file (config file), then AutoUpgrade uses the only SID in the configuration file. Specifying the SID is optional.
>
> However, if two or more SIDs are specified in the config file, then you must specify the SID for each database, then AutoUpgrade is unable to identify for which SID it should use the credentials. The result is an error.

Usage Notes

When you run AutoUpgrade for the database upgrade, AutoUpgrade reads the credential from the path specified by the local `wincredential`, so that AutoUpgrade is able to create services automatically in the Target database without requiring an administrator to provide Administrator credentials manually during the upgrade.

Be aware that the account used in your connection can be restricted by a local security policy so that it either is not allowed to log on, or the login may be denied. If that is the case, then you see an error such as the following:.

```
Start-Process : This command cannot be run due to the error:
Logon failure: the user has not been granted the requested
```

```
logon type at this computer.
At line:1 char:1
```

To work around that error, work with your Microsoft Windows administrator to add the user account running AutoUpgrade to the local policy setting:

1. Log on to the system (preferably as an administrator user).
2. Go to **Local Security Policy**.
3. Navigate to **Security Settings**, select **Local Policies**, and select **User Rights Assignment**.
4. For the following policies, verify that either the group to which the user running AutoUpgrade belongs or the user account itself is added to the following lists:
   - Access this computer from Network
   - Allow log on locally
   - Allow log on through Remote Desktop Services

For details, see Microsoft Support: "Logon type has not been granted."

Example

In the following example, you complete these steps in order to automate calling the Administrator credentials during the upgrade:

1. Create the configuration file, using the `wincredential` local parameter to specify the location for the Windows Powershell credential for the source database `db12201`:

```
global.autoupg_log_dir=C:\Users\oracle\autoupg
global.target.version=19.0.0
global.target_home=C:\u01\app\oracle\product\19\dbhome_1

upg1.sid=db12201
upg1.source_home=C:\u01\app\oracle\product\12.2\dbhome_1
upg1.log_dir=C:\Users\Oracle\autoupg
upg1.upgrade_node=localhost
upg1.target_base=C:\u01\app\oracle
upg1.target_version=19.0.0.0
upg1.wincredential=C:\Users\oracle\cred
```

2. Run AutoUpgrade in Configuration mode, using the `load_win_credential` command-line parameter. If there s only one SID in your configuration file, then specifying the system identifier (`SID`) is optional, but not required. For example:

```
C:\Users\oracle>java -jar autoupgrade.jar -config config.cfg -
load_win_credential db12201
AutoUpgrade 24.1.240306 launched with default internal options
Processing config file ...
```

> **Note:**
>
> If the config file contains multiple SIDs, then you must specify the SID when you run AutoUpgrade. If you do not specify the SID when there are multiple SIDs in the configuration file, then the result is an error.

3. A Microsoft Windows Powershell Credential prompt opens. Provide the credentials for the Oracle Database binary owner. PowerShell then generates the credential object (in this example, `db12201.xml`), and places it in the path that you specified with the `wincredential` parameter.
4. During the upgrade, run AutoUpgrade using the configuration file that specifies the credential object path. For example:

   ```
   C:\Users\oracle>java -jar autoupgrade.jar -config config.cfg -mode deploy
   ```

   AutoUpgrade processes the upgrade without prompting you for credentials.

Suppose your configuration file contains two SIDs, `db12201` and `db19x`:

```
upg1.sid=db12201
upg1.source_home=C:\databases\ee\product\12.2\dbhome_1
upg1.target_home=C:\databases\ee\product\18x\dbhome_1
upg1.wincredential=C:\Users\oracle\cred

upg2.sid=db19x
upg2.source_home=C:\databases\ee\product\18x\dbhome_1
upg2.target_home=C:\databases\ee\product\19x
upg2.wincredential=C:\Users\oracle\anothercred
```

In this case, running `java -jar autoupgrade.jar -config config.cfg -load_win_credential` without specifying the SID results in an error. When you run AutoUpgrade in Configuration mode to load credentials, you **must** provide the SID so that AutoUpgrade can load the credentials for that SID. For example, to load the correct credentials for `db12201`:

```
java -jar autoupgrade.jar -config config2.cfg -load_win_credential db12201
```

To load credentials for `db19x`, you conversely must specify the `db19x` SID:

```
java -jar autoupgrade.jar -config config2.cfg -load_win_credential db19x
```

Related Topics

- Microsoft Support: Logon type has not been granted

mode

The AutoUpgrade parameter `mode` value sets the mode from which AutoUpgrade runs.

Property	Description
Parameter type	string

Property	Description
Syntax	`-mode = [analyze|create_home|download|fixups|deploy|upgrade|postfixups]`
Default value	None. Choose one of the following options: • `analyze` : Runs upgrade readiness checks in the source Oracle home. • `create_home` : Used with AutoUpgrade patching only. When set, a new target Oracle home is created as part of the patching operation without requiring a source database. • `download` : Used with AutoUpgrade patching only. When set, AutoUpgrade downloads patches specified in the configuration file, and then exits. • `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. Note: If autoupgrade.jar is run with the -patch parameter, then the command cannot include the `-mode upgrade` option. • `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
```

Patch

The AutoUpgrade parameter `patch` is used to specify that AutoUpgrade is performing a patch operation, and not an upgrade operation.

Property	Description
Parameter type	String
Syntax	`-patch`
Default value	Not applicable

Description

The `patch` command-line parameter specifies that the autoupgrade.jar command starts an AutoUpgrade patching operation.

Usage Notes

This parameter is used only with AutoUpgrade patching. It cannot be used with the `-preupgrade` command-line option. All other command-line options can be used with the patch option.

The `-mode` operations analyze, fixup, deploy, download, and create_home are supported with the patch command-line parameter. The modes `download` and `create_home` are specifically for use with the patch parameter:

- `-mode download`: Specifies to download specified patches and then exit.
- `-mode create_home` Specifies to create the new target Oracle home without requiring a source database
- `-mode opatch` Specifies to use the latest version of OPatch.

Example

In this example, the patch parameter is used with `-mode download` to automatically download the patches specified in the configuration file:

```
java -jar autoupgrade.jar -patch -mode download
```

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 19c.

```
java -jar autoupgrade.jar -preupgrade "target_version=19" -mode analyze
```

Running AutoUpgrade with the preupgrade clause using `fixups` mode, and specifying that the target release is Oracle Database 19c.

```
java -jar autoupgrade.jar -preupgrade "target_version=19" -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\19.0.0\dbhome_1`.

```
java -jar autoupgrade.jar -preupgrade
"target_home=C:\app\oracle\product\19.0.0\dbhome_1" -mode postfixups
```

Running AutoUpgrade with the preupgrade clause without specifying the mode, and specifying that the target release is Oracle Database 19c. In this case, the mode used is the default mode, `analyze`.

```
java -jar autoupgrade.jar -preupgrade "target_version=19"
```

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.version 22.1.220304
build.date 2022/03/04 13:29:34 -0500
build.hash 29007da
build.hash_date 2022/03/04 12:48:36 -0500
build.supported_target_versions 12.2,18,19,21
build.type production
```

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] [-zip_exclusion_list exclusion_list]`
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. Use the `-zip_exclusion_list` clause to specify a double-comma-delimited regular string list that is used to exclude files that match any regular string from the zip file.

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
```

```
java -jar autoupgrade.jar -config yourconfig.cfg -zip -zip_exclusion_list ".*/
db11204/.*"
```

```
java -jar autoupgrade.jar -config yourconfig.cfg -zip -zip_exclusion_list "/home/
oracle/autopatch/DB19X/100/goldimage/db_home_2023-09-21_09-18-13AM.zip,,/home/
oracle/autopatch/DB19X/100/extract/35320081/.*"
```

AutoUpgrade Utility Configuration Files Parameters and Options

AutoUpgrade configuration files contain all the information required to perform Oracle Database upgrades.

Before you can use an AutoUpgrade processing mode, you must create an AutoUpgrade configuration file for the databases that you want to upgrade.

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.

- Locally Modifiable Global Parameters for AutoUpgrade Configuration File
  Required configuration parameters for AutoUpgrade can be set either globally for all upgrades, or locally.
- 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.
- 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.

Locally Modifiable Global Parameters for AutoUpgrade Configuration File

Required configuration parameters for AutoUpgrade can be set either globally for all upgrades, or locally.

Usage Notes

If you set required AutoUpgrade parameters globally, as a locally modifiable global parameter, then these parameters can be overridden by local parameters set for particular upgrades, so that you can better control AutoUpgrade job processing.

With locally modifiable global parameters, you can use the prefix `global` to set values for required parameters as global parameters for all jobs in your AutoUpgrade configuration file, but identify the same parameter with a local job prefix to reset the global value to a different value 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.

For example, with `global.target_home`, 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/19.0.0/dbhome01
upgrade3.target_home=/u03/app/oracle3/12.2.0.1/dbhome3
```

- defer_standby_log_shipping
- dictionary_stats_after
  (Optional) Specifies that AutoUpgrade gathers data dictionary statistics on the target database after the upgrade is complete.
- dictionary_stats_before
  (Optional) Specifies that AutoUpgrade gathers data dictionary statistics on the source database before starting the upgrade.
- drop_grp_after_upgrade
  Deletes the Guaranteed Restore Point (GRP) after database upgrade.

- enable_local_undo
  For a CDB upgrade, specifies whether or not `LOCAL` undo should be enabled before the upgrade of `CDB$ROOT`.
- export_rman_backup_for_noncdb_to_pdb
  (Optional) Specifies that AutoUpgrade transports metadata from the source non-CDB database to the target PDB database as part of the conversion process.
- fixed_stats_before
  (Optional) Specifies that AutoUpgrade gathers fixed object statistics on the source database before starting the upgrade.
- manage_network_files
  Specifies whether network files are processed during the upgrade.
- patch_in_upgrade_mode
  (Optional) Specifies that the database that you want to patch is patched in upgrade mode, instead of normal mode.
- remove_underscore_parameters
  Removes underscore (hidden) parameters from `PFILE` files during upgrade, and after upgrade, for all Oracle Databases in the configuration file.
- restoration
  (Available with Enterprise Edition only) Generates a Guaranteed Restore Point (GRP) for database restoration.
- rman_catalog_connect_string
  (Optional) Specifies the RMAN connection string used to connect to an RMAN database.
- target_base
  Specifies the target `ORACLE_BASE` path for the target Oracle home.
- target_home
  (**Required for upgrade and deploy modes, if the target home is not on the system**. Optional for analyze and fixups mode.) Specifies the target Oracle home (`ORACLE_HOME`) path.
- target_version
  (**Required if target Oracle home is not on the system, or is release 12.2**) Specifies the target release version on which you want AutoUpgrade to perform the upgrade.

defer_standby_log_shipping

Defers shipping logs from the primary database to any standby database. All log archive destionations (`log_archive_dest_n`) are set to deferred.

Usage Notes

By default, log shipping occurs as part of the upgrade. When Autoupgrade defers log shipping, you 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.

> **Note:**
>
> This configuration file parameter affects not only standby databases, but all products or services that receive redo from the primary database, such as Oracle Zero Data Loss Recovery Appliance (ZDLRA) real-time log transport, and Oracle GoldenGate downstream capture.

Options

`[yes | no]`

The default value is `no`

The default is no (log-shipping is not deferred). If you change the default to `Yes`, then log shipping is deferred, and you must choose to re-enable it manually after upgrade.

Example

```
defer_standby_log_shipping=yes
```

dictionary_stats_after

(Optional) Specifies that AutoUpgrade gathers data dictionary statistics on the target database after the upgrade is complete.

Usage Notes

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. When you specify yes, AutoUpgrade gathers dictionary statistics after the upgrade is completed.

Options

`[yes | no]`

The default value is `Yes`.

Example

```
global.dictionary_stats_after=yes


sales.dictionary_stats_after=yes
```

dictionary_stats_before

(Optional) Specifies that AutoUpgrade gathers data dictionary statistics on the source database before starting the upgrade.

Usage Notes

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. When you specify yes, AutoUpgrade gathers dictionary statistics before beginning the upgrade.

Options

`[yes | no]`

The default value is `Yes`.

Example

```
global.dictionary_stats_before=yes


sales.dictionary_stats_before=yes
```

drop_grp_after_upgrade

Deletes the Guaranteed Restore Point (GRP) after database upgrade.

Usage Notes

If you select this option, then GRP is deleted after upgrade completes successfully.

Options

`[yes | no]`

The default value is `no`.

Example

```
global.drop_grp_after_upgrade=yes


sales.drop_grp_after_upgrade=yes
```

enable_local_undo

For a CDB upgrade, specifies whether or not `LOCAL` undo should be enabled before the upgrade of `CDB$ROOT`.

Usage Notes

If you select this option, then AutoUpgrade runs the following statement before upgrade: `ALTER DATABASE LOCAL UNDO ON;`.

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.

Options

`[yes | no]`

The default value is `no`.

Example

```
enable_local_undo=yes
```

export_rman_backup_for_noncdb_to_pdb

(Optional) Specifies that AutoUpgrade transports metadata from the source non-CDB database to the target PDB database as part of the conversion process.

Usage Notes

When converting a non-CDB to a PDB, you can extract the RMAN metadata from the source database, and put it into the target database, so that the metadata it will be available after the PDB conversion. Using this parameter enables the backups to be used as "pre-plugin" backups. If you want to restore the PDB right after plug-in, then the pre-plugin backups option can help to save time and effort.

This parameter applies to non-CDB to PDB conversions only (not with refreshable clone PDBs). In all other cases, the parameter should be ignored.

Options

`[yes | no]`

The default value is `no`.

Example

```
sales.export_rman_backup_for_noncdb_to_pdb=yes
```

fixed_stats_before

(Optional) Specifies that AutoUpgrade gathers fixed object statistics on the source database before starting the upgrade.

Usage Notes

Before an upgrade, Oracle recommends that you regather fixed object statistics.

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 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.

Options

`[yes | no]`

The default value is `Yes`.

Example

```
global.fixed_stats_before=yes
```

```
sales.fixed_stats_before=yes
```

manage_network_files

Specifies whether network files are processed during the upgrade.

Usage Notes

If you select this option, then AutoUpgrade processes network files, depending on the option that you specify.

The following network files are processed: `oranfstab`, `ldap.ora`, `tnsnames.ora`, `sqlnet.ora`, and `listener.ora`

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

Example

```
manage_network_files=ignore_read_only
```

patch_in_upgrade_mode

(Optional) Specifies that the database that you want to patch is patched in upgrade mode, instead of normal mode.

Usage Notes

In AutoUpgrade 23.4 and earlier versions, the default for patching has been to perform patching in upgrade mode. Starting with AutoUpgrade 24.1, the default is to perform patching in normal mode. If you prefer to perform patching only in upgrade mode, then you can use this parameter to override that default behavior, and patch in upgrade mode.

Options

`[yes | no]`

The default value is `no`.

Example

```
sales.patch_in_upgrade_mode=yes
```

remove_underscore_parameters

Removes underscore (hidden) parameters from `PFILE` files during upgrade, and after upgrade, for all Oracle Databases in the configuration file.

Usage Notes

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
```

restoration

(Available with Enterprise Edition only) Generates a Guaranteed Restore Point (GRP) for database restoration.

Usage Notes

This option determines whether database backup and database restoration must be performed manually by the DBA.

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.

Options

`[yes | no]`

The default value is `yes`.

Example

```
global.restoration=no
```

rman_catalog_connect_string

(Optional) Specifies the RMAN connection string used to connect to an RMAN database.

Usage Notes

To use this feature, you must save the RMAN username and password in the keystore using the AutoUpgrade command-line parameter `load_password`.

Example

```
global.target_base=/u01/app/oracle
sales4.rman_catalog_connect_string=string-alias
```

Related Topics

- Daniel Overby Hansen Databases are Fun: AutoUpgrade New Features: Upgrade RMAN Catalog Schema

target_base

Specifies the target `ORACLE_BASE` path for the target Oracle home.

Example

```
global.target_base=/u01/app/oracle
sales4.target_base=/u04/app/oracle4
```

target_home

(**Required for upgrade and deploy modes, if the target home is not on the system**. Optional for analyze and fixups mode.) Specifies the target Oracle home (`ORACLE_HOME`) path.

Usage Notes

Use this option to specify the path to the target database home for the upgrade. This parameter can overwrite a global `target_home` setting.

```
sales1.target_home=/target/Oracle/home
```

Options

Earlier releases of AutoUpgrade required you to set `target_home` and `target_version`. In later releases of AutoUpgrade, this restriction has been lifted for both `Analyze` and `Fixups` modes. However, if you don't set `target_home`, then you must specify `target_version`. Either one of them must be present.

Example

```
sales3.target_home=/U01/app/oracle/product/19.0.0/dbhome_1
sales1.target_home=/U01/app/oracle/product/23.0.0/dbhome_1
```

target_version

(**Required if target Oracle home is not on the system, or is release 12.2**) Specifies the target release version on which you want AutoUpgrade to perform the upgrade.

Usage Notes

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.

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.

Options

Valid values

- 12.2
- 18
- 19
- 21
- 23

Example

```
global.target_version=23
employees.target_version=19
```

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. Local parameters that either must be set locally, or as a locally modifiable global parameter are indicated by (**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
```

- add_after_upgrade_pfile
  (Optional) Specifies a path and file name of a `PFILE` whose parameters you want to add after the upgrade.
- add_during_upgrade_pfile
  (Optional) Specifies a path and file name of a PFILE whose parameters you want to add during the upgrade.

- after_action
  (Optional) In `deploy` mode, specifies a custom action that you want to have performed after completing the deploy job for the database identified by the prefix address.
- before_action
  (Optional) In `deploy` mode, 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)
- 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.
- 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.
- close_source
  (Optional) Closes the source non-CDB or source PDB just before AutoUpgrade starts a non-CDB to PDB conversion, starts an unplug-relocate upgrade, or uses a refreshable clone PDB.
- del_after_upgrade_pfile
  (Optional) Specifies a path and file name of a `PFILE` whose parameters you want to have removed after upgrade.
- del_during_upgrade_pfile
  (Optional) Specifies a path and file name of a PFILE whose parameters you want to have removed during upgrade.
- delete_wincredential_file
  (Optional) Deletes the Microsoft Windows credential object file when the AutoUpgrade job is complete.
- download
  (Optional for AutoUpgrade patching) Specifies whether to automatically download patches from My Oracle Support. .
- drop_win_src_service
  (Optional) For upgrades on Microsoft Windows, specifies whether to drop the Windows operating system service for the source Oracle Database after upgrade.
- env
  (Optional) Specifies custom operating system environment variables set on your operating system, excluding `ORACLE_SID`, `ORACLE_HOME`, `ORACLE_BASE`, and `TNS_ADMIN`.
- exclusion_list
  (Optional) Sets a list of PDBs that you want to be excluded from the AutoUpgrade run. This parameter only applies to the multitenant architecture (CDB) databases. If you are plugging in and upgrading a non-CDB database, then this parameter is ignored.
- folder
  (Required for AutoUpgrade patching) Specifies the directory that contains the patch zip files as well as the required Oracle Database base image.
- home_settings.ru_apply
  (Optional) Specifies whether a release update (RU) is installed at the same time as the `ORACLE_HOME`, or installed as a separate step by OPatch.

- ignore_errors
  (Optional) Enables you to specify a comma-delimited list of specific Oracle errors that you want AutoUpgrade to ignore during the upgrade or patching process.
- keep_pdb_save_state
  (Optional) Specifies that if the PDB has a saved state in the source CDB, then you can either save or not save the PDB state on the target CDB.
- keep_source_pdb
  (Optional) Specifies if the source PDB in an unplug-plug upgrade operation is kept in a closed state instead of being removed from the source CDB.
- log_dir
  (Optional) 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.
- manage_standbys_clause
  (Optional) Specifies whether standby Oracle Data Guard standby databases you identify by `DB_UNIQUE_NAME` are excluded from AutoUpgrade plug-in upgrades, so that standby database files can be reused.
- method
  (Optional for AutoUpgrade patching) Specifies whether to create a new target Oracle home, and if so, how it will be created.
- parallel_pdb_creation_clause
  (Optional) Specifies the number of parallel execution servers to use when creating a pluggable database. This can be used for scenarios like PDB relocate, clone PDB, etc. It doesn't apply for cases when creating a PDB using an XML file.
- patch
  (Required for AutoUpgrade patching) Specifies a comma-delimited list of the patches that you want to install.
- patch_in_upgrade_mode
  (Optional) Specifies that the database that you want to patch is patched in upgrade mode, instead of normal mode.
- 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.
- platform
  (Optional) Specifies the platform used by AutoUpgrade Patching when downloading patches from My Oracle Support.
- raise_compatible
  (Optional) Increases the Oracle Database `COMPATIBLE` initialization parameter to the default value of the target release after the upgrade is completed successfully.
- remove_rac_config
  (Optional) Specifies whether to remove a non-CDB Oracle RAC database from clusterware on the source Oracle home after a successful conversion to the target CDB home, or to leave the source database unchanged.
- remove_underscore_parameters
  (Optional) Removes underscore (hidden) parameters from `PFILE` files during upgrade, and after upgrade, for all Oracle Databases in the configuration file.
- replay
  (Optional) Specifies whether to use replay to upgrade the database.

- restoration
  (Optional) Generates a Guaranteed Restore Point (GRP) for database restoration.
- revert_after_action
  (Optional) Specifies a custom action that you want to have run on the operating system after a system restoration is completed for the specific database job addressed by the prefix, and the database is up.
- revert_before_action
  (Optional) Specifies a custom action that you want to have run on the operating system before a system restoration is completed for the specific database job addressed by the prefix, and the database is up.
- run_dictionary_health
  (Optional) Specifies whether you run Oracle Dictionary Health Checks as part of preupgrade checks to identify database dictionary inconsistencies.
- run_utlrp
  (Optional) Enables or disables running a version of `utlrp.sql` as part of post upgrade, to recompile only invalid objects in Oracle-maintained schemas.
- sid
  (Required) Identifies the Oracle system identifier (SID) of the database that you want to upgrade.
- skip_tde_key_import
  (Optional) When set to `yes`, the upgrade is run, but import of the source database KeyStore into the target database is skipped, without raising an error.
- source_base
  (Optional) Specifies the source `ORACLE_BASE` path for the source Oracle home.
- source_dblink
  (Optional) Specifies the database link set up for an unplug-plug relocate (hot clone) upgrade.
- 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.
- source_ldap_admin_dir
  (Optional) Specifies the path to the `LDAP_ADMIN` directory in the source database home.
- source_tns_admin_dir
  (Optional) Specifies the path to the `TNS_ADMIN` directory in the source database home.
- 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.
- target_base
  (Optional) Specifies the target `ORACLE_BASE` path for the target Oracle home.
- target_cdb
  (Optional) Specifies the `SID` of the target CDB into which a non-CDB Oracle Database is plugged in.
- target_pdb_copy_option=file_name_convert
  (Optional) Specifies the `file_name_convert` option used by the create pluggable database statement that AutoUpgrade runs 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.

- target_pdb_name
  (Optional) Specifies the name that you want to assign to a non-CDB source Oracle Database after it is plugged in to the target CDB.
- target_ldap_admin_dir
  (Optional) Specifies the path to the `LDAP_ADMIN` directory in the target database home.
- target_tns_admin_dir
  (Optional) Specifies the path to the `TNS_ADMIN` directory in the target database home.
- timezone_upg
  (Optional) Enables or disables running the time zone upgrade as part of the AutoUpgrade process.
- tune_setting
  (Optional) Enables special workflows that alter the behavior of AutoUpgrade during runtime, depending on the workflow option that you specify.
- upgrade_node
  (Optional) Specifies the node on which the current user configuration is valid. The default value is `localhost`.
- wincredential
  (Optional) Specifies the location of a Microsoft Windows credential object file that you have previously generated with the AutoUpgrade command-line parameter `load_win_credential`.

add_after_upgrade_pfile

(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to add after the upgrade.

Examples

```
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.

Examples

```
sales3.add_during_upgrade_pfile=/path/to/my/newpfile.ora
```

after_action

(Optional) In `deploy` mode, specifies a custom action that you want to have performed after completing the deploy job for the database identified by the prefix address.

Usage Notes

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.

The output of the script is captured and stored in files. Both `stdout` and `stderr` are captured. The files are stored in the `postupgrade` subdirectory in the directory matching the specific database or job.

The following environment variables are set in the shell that runs the script:

- `ORACLE_SID`
- `ORACLE_UNQNAME`
- `ORACLE_BASE`
- `ORACLE_HOME`
- `TNS_ADMIN`

Examples

Run the specified script after 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 after AutoUpgrade starts processing the deploy option, with AutoUpgrade set to continue to run if the script fails:

```
sales3.after_action=/user/path/script.sh
```

before_action

(Optional) In `deploy` mode, 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)

Usage Notes

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.

The output of the script is captured and stored in files. Both `stdout` and `stderr` are captured. The files are stored in the `preupgrade` subdirectory in the directory matching the specific database or job.

The following environment variables are set in the shell that runs the script:

- `ORACLE_SID`
- `ORACLE_UNQNAME`
- `ORACLE_BASE`
- `ORACLE_HOME`
- `TNS_ADMIN`

Examples

Run the specified script before AutoUpgrade starts processing `deploy` mode, 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
```

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.

Usage Notes

Available `catctl.pl` options:

- `-n` Number of processes to use for parallel operations. For Replay upgrades, the number of parallel processes used for the upgrade defaults to the value of (`CPU_COUNT` divided by 4) . For Classic upgrades, the default for `CDB$ROOT` and `NON-CDB` databases is 8.
- `-N` Number of processors to use when upgrading PDBs. For Replay upgrades, the number of parallel processes used for the upgrade defaults to the value of (`CPU_COUNT` divided by 4) For Classic upgrades, the default is 2

- `-T` Takes offline user schema-based table spaces.
- `-z` Turns on production debugging information for `catcon.pl`.

Examples

```
sales4.catctl_options=-n 24 -N 4
```

Related Topics

- Upgrade Script (catctl.pl) Parameters

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.

Usage Notes

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.

Examples

```
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`.

close_source

(Optional) Closes the source non-CDB or source PDB just before AutoUpgrade starts a non-CDB to PDB conversion, starts an unplug-relocate upgrade, or uses a refreshable clone PDB.

Usage Notes

During the operations described above, if `close_source` is set to `yes` (the default), then AutoUpgrade closes source non-CDB or source PDB just before starting the upgrade. Additionally, if Oracle Real Application Clusters or Oracle Grid Infrastructure (CRS) services are configured for a non-CDB source, then they are disabled before starting the upgrade.

This parameter can only be used when the source and target databases are both on the same system. When they are on different systems, the source non-CDB or PDB cannot be closed, because AutoUpgrade has no access to it.

Options

`[yes | no]`

The default value is `yes`.

Examples

```
sales3.close_source=yes
```

del_after_upgrade_pfile

(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to have removed after upgrade.

Examples

```
sales3.del_after_upgrade_pfile=/path/to/my/pfile_del.ora
```

del_during_upgrade_pfile

(Optional) Specifies a path and file name of a PFILE whose parameters you want to have removed during upgrade.

Examples

```
sales3.del_during_upgrade_pfile=/path/to/my/oldpfile.ora
```

delete_wincredential_file

(Optional) Deletes the Microsoft Windows credential object file when the AutoUpgrade job is complete.

Usage Notes

When set to `NO`, AutoUpgrade does not delete the Microsoft Windows file credential after the AutoUpgrade job first using the credential is completed. The default value is `YES`.

The purpose of this parameter is to enable you to choose whether AutoUpgrade immediately deletes the Microsoft Windows object credentials loaded with the `wincredential` parameter after these credentials are first used, or if you want to be able to reuse the Windows object credential with other AutoUpgrade patching or upgrading operations.

> **Note:**
>
> If you set `delete_wincredential_file` to `NO`, then you must manually delete that credential after your AutoUpgrade jobs are complete. AutoUpgrade provides a notice in the postupgrade summary report to tell you that the Windows credential file was not removed, and that you should remove this credential file manually.

Use case:

You are performing multiple upgrades or patching operation with AutoUpgrade, and you want to specify credentials for the owner of database binaries on a Microsoft Windows server for more than one upgrade or patching operation so that they can all complete automatically. When you specify the `wincredential` parameter to load the credentials, and then also specify `delete_wincredential_file` to `NO`, AutoUpgrade can use that credential for multiple upgrades or patches of the same Oracle Database, or for different Oracle Databases. To use this feature, you must have already created the Windows PowerShell credential object, and then specify that credential object in the configuration file using `wincredential`.

Example

In the following example, the local configuration file setting `wincredential` provides the location where the Microsoft Windows credentials have been loaded, and `delete_wincredential_file=NO` specifies that AutoUpgrade does not automatically delete the Windows object credential file after the `db12201` database operation is complete.

```
global.autoupg_log_dir=C:\Users\oracle\autoupg
global.target.version=19.0.0
global.target_home=C:\u01\app\oracle\product\19\dbhome_1

upg1.sid=db12201
upg1.source_home=C:\u01\app\oracle\product\12.2\dbhome_1
upg1.log_dir=C:\Users\Oracle\autoupg
upg1.upgrade_node=localhost
upg1.target_base=C:\u01\app\oracle
upg1.target_version=19.0.0.0
upg1.wincredential=C:\Users\oracle\cred
upg1.delete_wincredential_file=NO
```

download

(Optional for AutoUpgrade patching) Specifies whether to automatically download patches from My Oracle Support. .

Usage Notes

Specifies whether to automatically download patches from My Oracle Support. The default is YES.

When set to YES, you must also load the My Oracle Support (MOS) credentials into AutoUpgrade Patching at the command line using the `-load_password` command-line option.

The patches that are downloaded are placed into the directory folder specified by the `folder` parameter.

If proxy information is required to connect to My Oracle Support, then set proxy values using the Linux operating system environment variables `https_proxy`, `http_proxy`, and `no_proxy`.

Options

```
[yes | no]
```

The default value is `yes`.

Examples

Override the default (yes) so that you use patches that you have downloaded manually instead of having AutoUpgrade download the patches automatically:

```
upg1.download=no
```

drop_win_src_service

(Optional) For upgrades on Microsoft Windows, specifies whether to drop the Windows operating system service for the source Oracle Database after upgrade.

Usage Notes

By default, for Oracle Database upgrades on Microsoft Windows operating systems, after AutoUpgrade shuts down the Windows Oracle Database service and completes the upgrade, it leaves the service in place. Leaving the service down but in place gives you the option to restore the database to the source Oracle home without having to recreate the Microsoft Windows service for the database. However, you can choose to have the Microsoft Windows service for the source database removed automatically after upgrade is completed successfully. If either `no` is specified, or no value is is specified, then the service is shut down on the source, but left in place after the upgrade.

Options

```
[yes | no]
```

The default value is `no`.

Examples

```
upg1.drop_win_src_service=yes
```

env

(Optional) Specifies custom operating system environment variables set on your operating system, excluding `ORACLE_SID`, `ORACLE_HOME`, `ORACLE_BASE`, and `TNS_ADMIN`.

Usage Notes

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. Multiple settings are comma-delimited.

Syntax:

prefix=VARIABLE1=value1 [, *VARIABLE2=value2*, ...]

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
```

exclusion_list

(Optional) Sets a list of PDBs that you want to be excluded from the AutoUpgrade run. This parameter only applies to the multitenant architecture (CDB) databases. If you are plugging in and upgrading a non-CDB database, then this parameter is ignored.

Usage Notes

Use this parameter to provide a list of PDBs to exclude from the AutoUpgrade run. The PDB list is comma-delimited. It can contain either a list of PDB names, or an asterisk character (*), which indicates that you want ot exclude all PDBs that are open on the CDB at the time that you run AutoUpgrade.

Syntax:

```
prefix.exclusion_list=[pdb-name|*][,pdb-name,...]
```

Examples

Assume that you want to exclude PDBs `pdb1` and `pdb2` from the upgrade of cdb sales1. The following entry in the configuration file excludes `pdb1` and `pdb2` from being processed during the AutoUpgrade run:

```
sales1.exclusion_list=pdb1,pdb2
```

This entry in the configuration file excludes all open PDBs from the CDB `sales2`:

```
sales2.exclusion_list=*
```

folder

(Required for AutoUpgrade patching) Specifies the directory that contains the patch zip files as well as the required Oracle Database base image.

Usage Notes

For AutoUpgrade to perform patching, you must identify the directory that contains the patch zip files. This directory must also contain the Oracle Database base image. There is no default value. You must provide the directory path. This parameter also is used in conjunction with the download parameter as follows:

When `download=YES`, the directory specified by the `folder` parameter is the directory to which the patches will be downloaded

When `download=NO`, the directory specified by the `folder` parameter is the directory that must contain the patches that have been manually downloaded.

The directory that you specify with the `folder` parameter must contain the base image of the source database's release (for example, Oracle Database Release 19.3).

Examples

```
upg1.folder=/storage/patches
```

home_settings.ru_apply

(Optional) Specifies whether a release update (RU) is installed at the same time as the `ORACLE_HOME`, or installed as a separate step by OPatch.

Usage Notes

Options: `YES` or `NO`. The default value is `NO`, unless the current operating system is Oracle Linux 9 or higher.

When the parameter is specified to `YES` on a platform other than Microsoft Windows, the RU being installed during a deploy operation is installed when `runInstaller` is run using the `-applyRU` command line option. When the parameter is specified to `NO`, the RU is then installed separately by running OPatch after the `ORACLE_HOME` has already been installed.

Examples

```
upg1.home_settings.ru_apply=true
```

ignore_errors

(Optional) Enables you to specify a comma-delimited list of specific Oracle errors that you want AutoUpgrade to ignore during the upgrade or patching process.

Usage Notes

If you add this parameter to your configuration file, then the error numbers that you specify are ignored during the upgrade for the upgrade prefix that you specify.

Examples

```
sales3.ignore_errors=ORA-48181,ORA-00001
```

keep_pdb_save_state

(Optional) Specifies that if the PDB has a saved state in the source CDB, then you can either save or not save the PDB state on the target CDB.

Usage Notes

This parameter applies to the Unplug-Plug flow (including restorations). If the PDB state is saved in the source CDB, then by default that same saved state continues to be saved after the upgrade process (default is `yes`). When `keep_pdb_save_state` is set to `no`, the source PDB state is not saved after the upgrade. You can choose to set `keep_pdb_save_state` to `no` when you are advised to do so in AutoUpgrade preupgrade checks. For example, with Oracle Real Application Clusters (Oracle RAC) upgrades, Oracle recommends that you do not keep the source PDB saved state.

Options

```
[yes | no]
```

The default value is `yes`.

Example

```
sales1.keep_pdb_save_state.pdbA=no
```

keep_source_pdb

(Optional) Specifies if the source PDB in an unplug-plug upgrade operation is kept in a closed state instead of being removed from the source CDB.

Usage Notes

By default, the source PDB is removed from the source CDB during the upgrade process. When `keep_source_pdb` is set to `YES`, the source PDB is not removed from the earlier release system. You are only able to set the parameter to `YES` when the copy option is specified in the parameter `target_pdb_copy_option`. When the copy option is not used, this parameter is ignored, because the PDB must be dropped. Without a copy, the existing datafiles can only be used by a single CDB.

Options

```
[yes | no]
```

The default value is `no`.

Example

```
sales1.keep_source_pdb=yes
```

log_dir

(Optional) 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.

Usage Notes

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.

> **Note:**
>
> On Microsoft Windows platforms, `global.autoupg_log` and `log_dir` should be configured on the same drive.

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.

manage_standbys_clause

(Optional) Specifies whether standby Oracle Data Guard standby databases you identify by `DB_UNIQUE_NAME` are excluded from AutoUpgrade plug-in upgrades, so that standby database files can be reused.

Usage Notes

Before upgrades of database configurations with standby databases, to reduce potential issues, Oracle recommends that you run AutoUpgrade in analyze mode on your standby databases.

Options

In the following syntax, *pdb-name* is a `DB_UNIQUE_NAME` of a source PDB that you are upgrading to the target CDB in an unplug/plug upgrade.

```
manage_standbys_clause=STANDBYS=[NONE|ALL|ALL EXCEPT ('pdb-name', 'pdb-
name', ...)|STANDBYS=('pdb-name', 'pdb-name', ...)]
```

The default value is `NONE`.

Examples

In the following example, any non-CDB or pluggable database that is a member of an Oracle Data Guard standby is not excluded from AutoUpgrade plug-in upgrades:

```
upg2.sid=cdb1
upg2.pdbs=*
upg2.target_cdb=cdb21x
upg2.source_home=/source/18x
upg2.target_home=/target/21x
upg2.manage_standbys_clause=standbys=none
```

In the following example, applying the redo on data files on all standby databases is deferred on all AutoUpgrade plug-in upgrades:

```
upg3.sid=cdb2
upg3.pdbs=*
upg3.target_cdb=cdb21x
upg3.source_home=/source/18x
upg3.target_home=/target/21x
upg3.manage_standbys_clause=standbys=all
```

In the following example, during the AutoUpgrade plug-in upgrades, applying the redo on data files is deferred on all standby PDBs except PDBs `cdb3_stby_1` and `cdb3_stby_2`.

```
upg4.sid=cdb3
upg4.pdbs=*
upg4.target_cdb=cdb21x
upg4.source_home=/source/12.2x
upg4.target_home=/target/21x
upg4.manage_standbys_clause=standbys=all except ('cdb3_stby_1','cdb3_stby_2')
```

In the following example, during the AutoUpgrade plug-in upgrades, applying the redo on data files is deferred only on standby PDB `cdb4_stby1`.

```
upg4.sid=cdb4
upg4.pdbs=*
upg4.target_cdb=cdb21x
upg4.source_home=/source/12.2x
```

```
upg4.target_home=/target/21x
upg4.manage_standbys_clause=standbys=('cdb4_stby_1')
```

method

(Optional for AutoUpgrade patching) Specifies whether to create a new target Oracle home, and if so, how it will be created.

Usage Notes

The default value is `outofplace`. At the time of this release, outofplace is the only permitted value. When the AutoUpgrade command-line parameter `-patch` is used, AutoUpgrade creates a new target Oracle home using the base image contained in the directory specified by the `folder` parameter. Oracle recommends that all patching is performed as out-of-place patching, where a new Oracle home is created.

Examples

```
upg1.method=outofplace
```

parallel_pdb_creation_clause

(Optional) Specifies the number of parallel execution servers to use when creating a pluggable database. This can be used for scenarios like PDB relocate, clone PDB, etc. It doesn't apply for cases when creating a PDB using an XML file.

Usage Notes

This parameter is optional. You can use this parameter to specify the number of parallel execution servers to copy the new PDB's data files to a new location when creating a PDB. This may result in faster creation of the PDB. Scenarios where you can take advantage of this option is when you want to upgrade and convert a non-CDB Oracle Database to a PDB, or you want to unplug a PDB from a source release CDB and relocate it in for an upgrade to a target release CDB.

The parameter is specific for every source database or pluggable database on the AutoUpgrade configuration file. The CDB can ignore this setting, depending on the current database load and the number of available parallel execution servers. Using this parameter enables you to provide better control of the load placed on the target database.

> **Note:**
>
> Note: This feature does not work during unplug/plug processes.

Options

Use an integer value to specify the number of servers to run in parallel, where *`source-db-name-or-pdb`* is the non-CDB database name or the PDB name, and *`integer-value`* is a numeric value specifying the number of servers to run in parallel::

`prefix``.parallel_pdb_creation_clause.`*`source-db-name-or-pdb`*`='integer-value'`

Example

In the following example, 16 servers are specified as the limit for the number of servers to run in parallel.

```
upg1.parallel_pdb_creation_clause.pdb1=16
```

patch

(Required for AutoUpgrade patching) Specifies a comma-delimited list of the patches that you want to install.

Usage Notes

Required for AutoUpgrade patching.

Options

`[recommended|ru|ru:x.y|opatch|ojvm|ojvm:x.y|dpbp|patch-number]`

The default value is `RECOMMENDED`.

Options:

- `RECOMMENDED`: Alias for all of the following options: `RU`, `OPATCH`, `OJVM`, `DPBP`.
- `RU`: Latest release update
- `RU:x.y`: A release update (RU) of the specified release version, where *x* is the major release number, and *y* is the RU. For example: `RU:19.24`
- `OPATCH`: Use latest version of OPatch
- `OJVM`: Apply the Oracle Java VM patch that applies to the specified RU.
- `OJVM:x.y`: Apply the Oracle Java VM patch of the specified release version, where *x* is the major release number, and *y* is the RU. For example: `OJVM:19.24`
- `DPBP`: Apply the Oracle Data Pump patch for the specified RU
- *patch-number*: specifies a specific one-off patch that you want AutoUpgrade to apply.

Examples

patch-number: Provide a specific one-off patch number.

```
upg1.patch=ru:19.24,12345678,opatch
```

Apply recommended patches, which includes the set of recommended patch options: `RU` (the latest release update for the base image), as well as `OPATCH`, `OJVM`, and `DPBP`:

```
upg1.patch=recommended
```

patch_in_upgrade_mode

(Optional) Specifies that the database that you want to patch is patched in upgrade mode, instead of normal mode.

Usage Notes

In AutoUpgrade 23.4 and earlier versions, the default for patching has been to perform patching in upgrade mode. Starting with AutoUpgrade 24.1, the default is to perform patching in normal mode. If you prefer to perform patching only in upgrade mode, then you can use this parameter to override that default behavior, and patch in upgrade mode.

Options

`[yes | no]`

The default value is `no`.

Example

```
sales.patch_in_upgrade_mode=yes
```

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.

Usage Notes

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 the parameter is not specified, then the default value is `*`.

If running in `ANALYZE` mode, AutoUpgrade ignores the PDBs in a mounted state.

If running in `FIXUPS`, `DEPLOY` or `UPGRADE` mode, AutoUpgrade opens the PDBs in mount state in read-write mode, upgrade mode, or both, depending on the execution mode.

Example

```
sales1.pdbs=pdb1, pdb2, pdbn
   upgr1.pdbs=*
```

platform

(Optional) Specifies the platform used by AutoUpgrade Patching when downloading patches from My Oracle Support.

Usage Notes

The parameter `platform` specifies which platform patches AutoUpgrade uses for patching. AutoUpgrade Patching supports the following platforrms:

- `AIX.x64` IBM AIX on POWER Systems (64-Bit)

- `ARM.x64` LINUX ARM (`aarch64`)
- `LINUX.X64` Linux x86-64
- `SPARC.x64` Oracle Solaris on SPARC (64-Bit)
- `SOLARIS.x64` Oracle Solaris on x86-64 (64-Bit)
- `WINDOWS.X64` Microsoft Windows x64 (64-bit).

If the current operating system is one of the supported platforms, then the default value matches that platform. Otherwise, the default value is `LINUX.X64`

Options

`[AIX.x64|ARM.x64|LINUX.X64|SPARC.x64|SOLARIS.x64|WINDOWS.X64]`

The default value is `LINUX.X64`.

Example

```
upg1.platform=LINUX.X64
```

raise_compatible

(Optional) Increases the Oracle Database `COMPATIBLE` initialization parameter to the default value of the target release after the upgrade is completed successfully.

Usage Notes

Options:

- `Y` : Increase the `COMPATIBLE` parameter setting to the target release
- `N` : Do not increase the `COMPATIBLE` parameter setting to the target release
- Increase the COMPATIBLE level to a specific release update (RU) level (for example, `23.0`, `23.4`, `23.7`)

The default is `N`.

> **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 specify to raise `COMPATIBLE` to a target RU level, then the the RU level you specify cannot be greater than the target Oracle home release. For example, if the target Oracle home release is Oracle Database 23ai updated to RU 23.4, then 23.7 is not a valid value for `raise_compatible`.

Examples

Raise `COMPATIBLE` to the level of the target Oracle Database home:

```
sales1.raise_compatible=yes
```

Raise `COMPATIBLE` to RU 23.4:

```
sales1.raise_compatible=23.4
```

Raise `COMPATIBLE` to RU 23.7:

```
sales1.raise_compatible=23.7
```

remove_rac_config

(Optional) Specifies whether to remove a non-CDB Oracle RAC database from clusterware on the source Oracle home after a successful conversion to the target CDB home, or to leave the source database unchanged.

Usage Notes

By default, the source Oracle RAC database configuration on a non-CDB is removed from the source Oracle Grid Infrastructure when it is migrated to a CDB during the upgrade process. When `remove_rac_config` is set to `no`, the source Oracle RAC database is not removed from the earlier release non-CDB system.

Options

```
[yes | no]
```

The default value is `yes`.

Example

```
upg1.remove_rac_config=no
```

remove_underscore_parameters

(Optional) Removes underscore (hidden) parameters from `PFILE` files during upgrade, and after upgrade, for all Oracle Databases in the configuration file.

Usage Notes

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
```

replay

(Optional) Specifies whether to use replay to upgrade the database.

Usage Notes

By default, AutoUpgrade performs a Classic upgrade to upgrade the database.

Options

`[yes | no]`

The default value is `no`.

Example

```
upg1.replay=yes
```

restoration

(Optional) Generates a Guaranteed Restore Point (GRP) for database restoration.

Usage Notes

If left at default value (`yes`), a fast recovery area configuration is required to create a GRP. If you set `restoration=no`, then both the database backup and restoration must be performed manually. Set to `no` for databases that operate in `NOARCHIVELOG` mode, and for Standard Edition and Standard Edition 2 databases, which do not support the Oracle Flashback technology feature Flashback Database. If you do not specify the parameter, then the default value (`yes`) is used, and a guaranteed restore point is created.

Options

`[yes | no]`

The default value is `yes`.

Example

```
sales1.restoration=no
```

revert_after_action

(Optional) Specifies a custom action that you want to have run on the operating system after a system restoration is completed for the specific database job addressed by the prefix, and the database is up.

Usage Notes

The action that you specify with `revert_after_action` runs on the target Oracle home binaries after the restoration process is completed, and the database is up.

The script that you specify to run 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 script (`.sql`), with a local operation on the database designated by the `revert_before_action` parameter prefix.

Options

Stop on fail: [`Y|N`]. The default is N.

By default, if the specified script fails, then AutoUpgrade continues to run (`N`. To specify that AutoUpgrade stops if the script fails, use the `Y` flag. If the script finishes running on the operating system with a status different than `0`, then AutoUpgrade identifies the script as failed.

Examples

Run the script you specify on the operating system after AutoUpgrade completes processing the restoration, with the `Y` flag set to stop AutoUpgrade if the script fails:

```
sales3.revert_after_action =/user/path/script.sh Y
```

Run the script you specify on the operating system after AutoUpgrade completes processing the restoration. With no flag, the default stop on fail option is `N`, so AutoUpgrade continues to run if the script fails:

```
sales3.revert_after_action =/user/path/script.sh
```

revert_before_action

(Optional) Specifies a custom action that you want to have run on the operating system before a system restoration is completed for the specific database job addressed by the prefix, and the database is up.

Usage Notes

The action that you specify with `revert_before_action` runs on the target Oracle home binaries before database restoration is started, and the database is up.

The script that you specify to run 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 script (`.sql`), with a local operation on the database designated by the `revert_before_action` parameter prefix.

Options

Stop on fail: [`Y|N`]. The default is `N`.

By default, if the specified script fails, then AutoUpgrade continues to run (`N`. To specify that AutoUpgrade stops if the script fails, use the `Y` flag. If the script finishes running on the operating system with a status different than `0`, then AutoUpgrade identifies the script as failed.

Examples

Run the script you specify on the operating system before AutoUpgrade starts the restoration, with the `Y` flag set to stop AutoUpgrade if the script fails:

```
sales3.revert_before_action =/user/path/script.sh Y
```

Run the script you specify on the operating system before AutoUpgrade starts the restoration. With no flag, the default stop on fail option is `N`, so AutoUpgrade continues to run if the script fails:

```
sales3.revert_before_action =/user/path/script.sh
```

run_dictionary_health

(Optional) Specifies whether you run Oracle Dictionary Health Checks as part of preupgrade checks to identify database dictionary inconsistencies.

Usage Notes

To help to identify database dictionary inconsistencies, you can specify that AutoUpgrade runs the `DBMS_DICTIONARY_CHECK` PL/SQL package on the source database as part of preupgrade checks. If set, the AutoUpgrade `run_dictionary_health` parameter enables you to specify for each upgrade source database that AutoUpgrade runs either the full array of Oracle Dictionary

Health Checks on the database dictionary, or that it runs only the most critical set of checks. If the check detects potential or critical problems with the database dictionary, then it prevents the upgrade from starting.

Oracle Dictionary Health Check results are stored under the AutoUpgrade precheck directory in the format *dbname*_healthcheck_result.log, where *dbname* is the name of the database on which the check was run. For more information about Oracle Dictionary Health Check, refer to the DBMS_HCHECK package documentation in *Oracle Database PL/SQL Packages and Types Reference*.

Options

```
[full| critical]
```

If the parameter is not set, then the default is to not run DBMS_DICTIONARY_CHECK.

Example

```
sales1.run_dictionary_health=full
sales2.run_dictionary_health=critical
```

Related Topics

- DBMS_DICTIONARY_CHECK PL/SQL Package

run_utlrp

(Optional) Enables or disables running a version of utlrp.sql as part of post upgrade, to recompile only invalid objects in Oracle-maintained schemas.

Usage Notes

The utlrp utility recompiles all Data Dictionary objects that become invalid during a database upgrade. If you set run_utlrp=no, or if you want invalid user objects also to be recompiled, then Oracle recommends that you use this utility to recompile invalid objects after upgrading with AutoUpgrade.

> **Note:**
>
> Starting with AutoUpgrade 23.1, when you run the AutoUpgrade utility, AutoUpgrade runs the utlprpom.sql script, and does not run utlrp.sql. When AutoUpgrade is used for upgrades to Oracle Database 12c Release 2 (12.2.0.1) and later releases, AutoUpgrade only recompiles invalid objects owned by Oracle-maintained schemas. Because database upgrades do not need to touch user objects, AutoUpgrade maintains this policy when it recompiles invalid objects.

Options

```
[yes | no]
```

The default value is yes.

Example

```
prefix.run_utlrp=yes
```

sid

(Required) Identifies the Oracle system identifier (SID) of the database that you want to upgrade.

Usage Notes

This parameter is case-sensitive.

Example

```
sales1.sid=salesdb
```

skip_tde_key_import

(Optional) When set to `yes`, the upgrade is run, but import of the source database KeyStore into the target database is skipped, without raising an error.

Usage Notes

> **Note:**
>
> This parameter is deprecated, because it is no longer needed. It can be removed in a future AutoUpgrade release. Instead of using this parameter, Oracle recommends that you either use the `-load_password` command line option to add the TDE password to AutoUpgrade's keystore, or add the TDE password to a Secure External Password Store (SEPS).

You can use this option for nonCDB-to-PDB and unplug/plug operations. When import of the source database KeyStore into the target database is skipped, AutoUpgrade will leave the PDB open in upgrade mode, so that you can import the keys manually yourself. After you import the keys, you must then restart the database in normal mode.

Options

```
[yes | no]
```

The default value is `no`.

Example

```
sales1.skip_tde_key_import=yes
```

source_base

(Optional) Specifies the source `ORACLE_BASE` path for the source Oracle home.

Examples

```
source_base=/u01/app/oracle
sales4.source_base=/u04/app/oracle4
```

source_dblink

(Optional) Specifies the database link set up for an unplug-plug relocate (hot clone) upgrade.

Usage Notes

To set up an unplug-plug relocate upgrade for a non-CDB or a PDB, you must first set up a database link between the source database and the target database location. You then pass that database link to AutoUpgrade using the `source_dblink` parameter. You identify source database name associated with the database link as a suffix to `source_dblink`. parameter. You also have the option to specify a time value, in seconds, that the database is refreshed from the database link.

> **Note:**
>
> This option is available for source database releases Oracle Database 12.1.0.2 and later.

The `source_dblink` parameter becomes active when you use the `target_pdb_copy_option` parameter. When you use `source_dblink`, you must also must specify a value for the `file_name_convert` parameter, either to convert file names, or to specify not to convert file names. If `file_name_convert` is set to `none`, then you must also set `db_create_file_dest` to specify where you want to place the database files.

You can also choose to set a refresh interval, in seconds, specifying how frequently the target database is updated over the database link from the source database. You can use the refresh interval with the `start_time` parameter to keep the source database refreshed for the target location. If no refresh rate is specified, then the source database is cloned only one time, and no refresh occurs. If the refresh rate is specified, but you do not specify a future start time using the `start_time` parameter, then the refresh interval value is ignored, and the database is cloned only one time.

Options

- (Required) The source database name, specified as a suffix.
- (Required) The name of the database link that you created.
- (Optional) The refresh rate for the target database from the source database, in seconds. If you specify a refresh rate, then typically you also specify a future start time using the `start_time` parameter.
- (Optional) `CLONE_ONLY`. Adding this option specifies that the PDB that is created is a clone that is never refreshed, and that the upgrade is started immediately after the clone

operation is completed. This option is required when the source is Oracle Database 12.1 (Release 12.1.0.2).

Examples

In the following example, two database links are created:

- `pdbxcdb18x_link`, created on the PDB source database named `pdbx`:

```
CREATE DATABASE LINK pdbxcdb18x_link CONNECT TO remote-user IDENTIFIED BY
password
USING'(DESCRIPTION =(ADDRESS = (PROTOCOL = TCP)(HOST
GRANT CREATE SESSION, CREATE PLUGGABLE DATABASE, SELECT_CATALOG_ROLE TO
remote-user;
GRANT READ ON sys.enc$ TO remote-user;
```

- `db18x_link`, created on the non-CDB source database named `db18x`:

```
CREATE DATABASE LINK db18x_link CONNECT TO remote-user IDENTIFIED BY
password
USING'(DESCRIPTION =(ADDRESS = (PROTOCOL = TCP)(HOST = db-node1)(PORT =
1521))
(CONNECT_DATA = (SERVICE_NAME = db18x)))';
```

In the AutoUpgrade configuration file, the database name associated with the database link is specified by using that name as a suffix to `source_dblink`: The suffix: `pdbx` for the PDB source database, and the suffix `db18x` for the non-CDB source database.

In the following example, `source_dblink` is used to specify the dblink for the source database `pdbx`. The PDB upgrade deployment starts immediately after you start AutoUpgrade, because no time interval is specified:

```
upg1.source_dblink.pdbx=pdbxcdb18x
```

Using the same configuration file, AutoUpgrade starts the upgrade of the database named `db18x` in 1 hour and 40 minutes after AutoUpgrade is started from the command line. Between the time that AutoUpgrade is started, and the deployment time specified by `start_time`, the cloned target database is refreshed every 20 seconds from the source.

```
upg1.source_dblink.db18x=db18x_link 20
upg1.start_time=+1h40m
```

In the following example, the source database `db18x` is cloned to the target PDB `db18x_link`, and the upgrade is started immediately after that source database is cloned successfully:

```
upg1.source_dblink.db18x=db18x_link CLONE_ONLY
```

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.

Usage Notes

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_ldap_admin_dir

(Optional) Specifies the path to the `LDAP_ADMIN` directory in the source database home.

Usage Notes

This parameter has no effect on Microsoft Windows, because on Windows, the `LDAP_ADMIN` environmental variable is set within the registry.

Example

```
sales1.source_ldap_admin_dir=/u01/app/oracle/12.2/dbhome01/ldap/admin
```

source_tns_admin_dir

(Optional) Specifies the path to the `TNS_ADMIN` directory in the source database home.

Usage Notes

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
```

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.

Usage Notes

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`.

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 that use the wrong date or hour format, such as the following:

```
30-12-2019 15:30:00
01/11/2020 3:30:15pm
2020/06/01 01:30:15
```

Examples

```
sales1.start_time=now
sales2.start_time=07/11/2020 01:30:15
```

target_base

(Optional) Specifies the target `ORACLE_BASE` path for the target Oracle home.

Examples

```
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.

Usage Notes

This parameter is mandatory when you want to upgrade and convert a non-CDB Oracle Database. It is case-sensitive.

Example

```
emp.target_cdb=salescdb
```

target_pdb_copy_option=file_name_convert

(Optional) Specifies the `file_name_convert` option used by the create pluggable database statement that AutoUpgrade runs 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.

Usage Notes

> **Caution:**
>
> Specifying `target_pdb_copy_option` enables AutoUpgrade to manage the recovery as needed. When `target_pdb_copy_option` is not set, and the default `nocopy` option is used, there is no recovery of the default PDB. Ensure that you have a backup of your source PDB.

This option is only used when creating a pluggable database within the target CDB. If you do not specify this parameter, then the default value of the parameter is `NOCOPY`, and existing data files on the source database are reused. When you do specify this parameter, then you must append a suffix to the parameter that specifies either the source database name or PDB name (`target_pdb_copy_option.`*`suffix`*, and specify `file_name_convert=` with one of the following options:

- Specify source file names (f) and target replacement file names (r) ('*f*', '*r*'), or specify `NONE`
- If you are creating a refreshable clone database, then append a suffix to the parameter that specifies either the source database name or PDB name (`target_pdb_copy_option.`*`suffix`*

On the target CDB, if you are using ASM, or 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 for replacement file names, then set the value of *`prefix`*`.target_pdb_copy_option.`*`source-db-name-or-pdb`*`=file_name_convert=NONE`.

If you want one or more data file names changed during conversion on the target CDB, then enter values for the parameter to indicate the source database name or PDB, specified as a suffix, the source filename you want to change, and the target filename to which you want the existing files copied, using the syntax *`prefix`*`.target_pdb_copy_option.`*`source-db-name-or-pdb`*`=('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.

Syntax

`prefix``.target_pdb_copy_option.`*`source-db-name-or-pdb`*`=file_name_convert=('f1', 'r1', 'f2', 'r2', 'f3', 'r3'...)`

Example

In this example, AutoUpgrade will copy existing datafiles during conversion of a database specified with the prefix string `upg1`, and with the suffix `sales` 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.sales=file_name_convert=('/old/path/pdb_2', '/new/
path/depsales')
```

To convert OMF files with `target_pdb_copy_option`*`source-db-name-or-pdb`*`=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.sales=file_name_convert=('+DATA/dbname/sales',
'+DATA/dbname/depsales')
```

target_pdb_name

(Optional) Specifies the name that you want to assign to a non-CDB source Oracle Database after it is plugged in to the target CDB.

Usage Notes

This parameter is optional. It is used when you want to upgrade and convert a non-CDB Oracle Database to a PDB, or you want to unplug a PDB from a source release CDB and plug it in for an upgrade to a target release CDB.

When you upgrade and convert an existing non-CDB database to a PDB on a target CDB, the `target_cdb` parameter is mandatory, because it specifies the target CDB. If you want to determine how the PDB is created on the target CDB, you can use the optional parameters `target_pdb_name` and `target_pdb_copy_option` to specify how the PDB is created on the target CDB. However, if neither optional parameters are used, then a full upgrade of the source CDB is performed.

The default name for the target PDB when you convert a non-CDB to a PDB is to use the database unique name of the non-CDB Oracle Database. To specify a name that is different from the existing name of the non-CDB when you plug it in to the CDB, set the new name by using `target_pdb_name`. In addition, if you are creating a refreshable clone database, then append a suffix to the parameter that specifies either the source database name or PDB name (*`target_name.suffix`*)

Examples

In the following example, the source non-CDB database is `emp19`. The target_pdb_name parameter is used to change the name to `emp23pdb` on the target CDB database.

```
upg.target_pdb_name=emp23pdb
```

For a refreshable clone, add a prefix to indicate the source database for the clone. In this example, the source container database is `db122b` and we are cloning `pdb1` from `db122b` into

the target container database `db19`. The suffix `pdb1` is used as the identifier for both `target_pdb_name` and `source_dblink`. The `pdb1` suffix identifier associates both the target PDB name and the dblink used to move the data from the source, `pdb1`, into the target PDB `PLUG122`.

```
global.autoupg_log_dir=/tmp/logs
upg1.source_home=/u01/app/oracle/122
upg1.target_home=/u01/app/oracle/19
upg1.sid=db122b
upg1.target_cdb=db19
upg1.pdbs=pdb1
upg1.target_pdb_name.pdb1=PLUG122
upg1.target_pdb_copy_option.pdb1=file_name_convert=('/u01/app/oracle/oradata/
db122b/pdb1', '/u01/app/oracle/plug/pdb122b')
upg1.source_dblink.pdb1=pdbxcdb122x_link
```

target_ldap_admin_dir

(Optional) Specifies the path to the `LDAP_ADMIN` directory in the target database home.

Example

```
sales1.target_ldap_admin_dir=/u01/app/oracle/19/dbhome01/ldap/admin
```

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.

Usage Notes

To preserve data integrity, Oracle recommends that you upgrade the time zone file (`DST`) 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.

If you explicitly disable the time zone file upgrade in your AutoUpgrade configuration file, then Oracle recommends that you perform this task either as part of your upgrade plan, or at a later point in time.

Options

`[yes | no]`

The default value is `yes` for upgrade, and `no` for patching.

Example

```
sales1.timezone_upg=no
```

> **Note:**
>
> If you patch a database with RU 19.18 or later, then updated time zone files are installed in the Oracle home by default. A new database created with Database Configuration Assistant (DBCA) in a patched Oracle home will be created with the latest time zone files.

tune_setting

(Optional) Enables special workflows that alter the behavior of AutoUpgrade during runtime, depending on the workflow option that you specify.

Usage Notes

The `tune_setting` parameter enables you to fine-tune upgrade steps or the resources allocated to the processing of the upgrades specified by the container databases or pluggable databases (CDBs or PDBs) specified by the parameter prefix in your AutoUpgrade configuration file. This capability can be useful for some upgrades if you find the default AutoUpgrade values are insufficient for your system requirements, or when you want to enable nondefault AutoUpgrade options.

Syntax

prefix`.tune_setting=`*option*`[, option, option, ...]`

Select the `tune_setting` options that provide the AutoUpgrade runtime tuning that you require from the list that follows. To combine multiple tuning options with the `tune_setting` parameter, use comma delimiters. Example:

```
sales3.tune_setting=proactive_fixups=true,query_hint_parallel=8,utlrp_threads_
per_pdb=8
```

> **Note:**
>
> You can concatenate multiple parameters together in a single `tune_setting` entry

Option	Description
`active_nodes_limit`	Sets a new total of active cluster member nodes that you want to use during a distributed upgrade of Oracle Real Application Clusters databases. The default is 2. If the number you specify is equal to or greater than the maximum number of cluster member nodes, then all nodes are taken. `sales3.tune_setting=active_nodes_limit=4`

Option	Description
`distributed_upgrade`	Specifies that AutoUpgrade performs a **distributed upgrade**. A distributed upgrade leverages the resources of the Oracle Clusterware cluster member nodes to perform the upgrades of PDBs more rapidly on the cluster. Use this option when a CDB in an Oracle RAC cluster of at least two nodes is being upgraded. When you choose this option, the `proactive_fixups` option is also enabled by default. Example: `sales3.tune_setting=proactive_fixups=true,distributed_upgrade=true` Note: Distributed upgrades are not supported on Microsoft Windows.
`make_pdbs_available`	Opens the PDBs designated by the prefix in read/write and non-restricted mode after postfixups are complete when **proactive fixups** mode is used. This option enables PDBs designated by the prefix to become available for service immediately after the upgrade is completed, while other PDBs continue to be upgraded, which can be useful for large fleet upgrade deployments. Precautions: Choosing this option enables the PDBs you designate to accept service requests from users, while other PDBs are being upgraded. The response time of the PDBs for service requests, and the time required for ongoing PDB upgrades can each be affected. Example: `sales3.tune_setting=make_pdbs_available=true`
`proactive_fixups`	Enables proactive fixups mode, where the PDBs are upgraded as the last stage of the upgrade. When the number of PDBs is higher than the CPU count defined in the database, divided by 2, choosing this tuning option can result in a faster upgrade. Example: `sales3.tune_setting=proactive_fixups=true` Precautions: If the number of CPUs is higher than the number of PDBs, then changing this setting may not improve performance.

`query_hint_parallel`	Specifies a parallel thread specification to the code that gathers data from the tablespaces during the query of the PDBs specified by the prefix, so that you can allocate a greater number or lesser number of parallel threads to the PDBs specified by the prefix. Example:
	`sales3.tune_setting=query_hint_parallel=8`
	Choosing this option can cause AutoUpgrade to consume more system resources.
	Option `NO_HINT` avoids the use of optimizer hints on the query that gathers information about database tablespaces. This option can be useful in environments where existing hints (materialize and parallel) affect the performance of the query. For example:
	`sales3.tune_setting=query_hint_parallel=NO_HINT`

Option	Description
utlrp_threads_per_pdb	Overwrites default maximum number of threads generated by the recompilation of invalid objects in the CDB, and uses the number of threads that you specify. Example: sales3.tune_setting=utlrp_threads_per_pdb=8 Precautions: If the number of threads specified exceeds available threads on the system, then performance can be compromised.
utlrp_pdb_in_parallel	Overwrites default maximum number of concurrent recompilation threads to the number that you specify. Use this option to overwrite the default maximum number of concurrent processes of recompilation of invalid objects. Example: sales3.tune_setting=utlrp_pdb_in_parallel=2 Precautions: Each PDB process requires from the system as many threads as specified by utlrp_threads_per_pdb.

Examples

In the following example, the database upgrades specified with the prefix sales3 are Oracle Real Application Clusters Oracle Database instances. The tune_setting parameter is used to set these database instances to use the setting distributed_upgrade, which distributes the upgrade load across multiple CDBs in the Oracle Grid Infrastructure cluster:

```
sales3.tune_setting=distributed_upgrade=true
```

In the following example, the database upgrades specified with the prefix sales3 are tuned with multiple tune_setting parameter options:

```
sales3.tune_setting=proactive_fixups=true,query_hint_parallel=8,utlrp_threads_
per_pdb=8
```

upgrade_node

(Optional) Specifies the node on which the current user configuration is valid. The default value is localhost.

Usage Notes

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
```

wincredential

(Optional) Specifies the location of a Microsoft Windows credential object file that you have previously generated with the AutoUpgrade command-line parameter `load_win_credential`.

Usage Notes

The purpose of this parameter is to create a credentials file to store the user and password credentials for the owner of the Oracle database binaries, and to specify the location of the Administrator PowerShell credential object for those credentials, so that AutoUpgrade can be run using that that credential object during the Oracle Database upgrade. To use this feature, you must have already created the Windows PowerShell credential object, and then specify that credential object in the configuration file using `wincredential`.

Use case:

You want to specify credentials for the owner of database binaries on a Microsoft Windows server. To specify these credentials, after you enter the wincredential paramter in your configuration file, you run AutoUpgrade in Configuration mode using the `load_win_credentials` command-line paramter, and provide credentials as prompted. Microsoft Window Powershell then creates the credential object, and stores the generated credential object in the path location you specify with `wincredential`. For example, in the following file, the location of the credential file is specified with `upg1.wincredential=C:\Users\oracle\cred`

Example

```
global.autoupg_log_dir=C:\Users\oracle\autoupg
global.target.version=19.0.0
global.target_home=C:\u01\app\oracle\product\19\dbhome_1

upg1.sid=db12201
upg1.source_home=C:\u01\app\oracle\product\12.2\dbhome_1
upg1.log_dir=C:\Users\Oracle\autoupg
upg1.upgrade_node=localhost
upg1.target_base=C:\u01\app\oracle
upg1.target_version=19.0.0.0
upg1.wincredential=C:\Users\oracle\cred
```

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, except for target_home when using upgrade or deploy mode. 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

- 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.
- add_during_upgrade_pfile
  (Optional) Specifies a path and file name of a `PFILE` whose parameters you want to add during the `PFILE` is upgraded.
- 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.
- autoupg_log_dir
  (Optional) Sets the location of the log files, and temporary files that belong to global modules, which AutoUpgrade uses.
- before_action
  (Optional) Specifies a custom user script that you want to have run for all upgrades before starting the upgrade jobs.
- 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.
- 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.
- 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.
- drop_grp_after_upgrade
  (Optional) Deletes the Guaranteed Restore Point (GRP) after database upgrade.

- global_log_dir
  (Optional for AutoUpgade Patching) Sets the location of the AutoUpgrade patching log files, and temporary files that belong to global modules, which AutoUpgrade uses.
- json_progress_writing_interval
- keystore
- raise_compatible
  (Optional) Increases the compatible parameter to the default value of the target release after the upgrade is completed successfully.
- replay
  (Optional) Specifies whether to use replay to upgrade the database.
- target_base
  (Optional) Specifies the target `ORACLE_BASE` path for the target Oracle home.
- 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.
- target_version
  (Optional) Specifies the target release version on which you want AutoUpgrade to perform the upgrade.
- upgradexml
  (Optional) Generates the `upgrade.xml` file.

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.

Usage Notes

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 add during the `PFILE` is upgraded.

Usage Notes

This specification applies to all databases in the user configuration file.

Example

```
global.add_during_upgrade_pfile=/path/to/my/add_during.ora
```

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.

Usage Notes

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.

The output of the script is captured and stored in files. Both `stdout` and `stderr` are captured. The files are stored in the `postupgrade` subdirectory in the directory matching the specific database or job.

The following environment variables are set in the shell that runs the script:

- `ORACLE_SID`
- `ORACLE_UNQNAME`
- `ORACLE_BASE`
- `ORACLE_HOME`
- `TNS_ADMIN`

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
```

autoupg_log_dir

(Optional) Sets the location of the log files, and temporary files that belong to global modules, which AutoUpgrade uses.

Usage Notes

You can configure different log directory path in the `userconfig` file in the logs directory for a specific prefix

This parameter cannot be used with AutoUpgrade Patching.

> **Note:**
>
> On Microsoft Windows platforms, `global.autoupg_log` and `log_dir` should be configured on the same drive.

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.

Examples

```
global.autoupg_log_dir=/path/to/my/global/log/dir
```

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.global_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
```

before_action

(Optional) Specifies a custom user script that you want to have run for all upgrades before starting the upgrade jobs.

Usage Notes

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.

The output of the script is captured and stored in files. Both `stdout` and `stderr` are captured. The files are stored in the `preupgrade` subdirectory in the directory matching the specific database or job.

The following environment variables are set in the shell that runs the script:

- ORACLE_SID
- ORACLE_UNQNAME
- ORACLE_BASE
- ORACLE_HOME
- TNS_ADMIN

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.

Options

Available `catctl.pl` options:

- `-n` Number of processes to use for parallel operations. For Replay upgrades, the number of parallel processes used for the upgrade defaults to the value of (`CPU_COUNT` divided by 4) . For Classic upgrades, the default for `CDB$ROOT` is 8, and the default for non-CDB databases is 8.
- `-N` Number of processors to use when upgrading PDBs. For Replay upgrades, the number of parallel processes used for the upgrade defaults to the value of (`CPU_COUNT` divided by 4) For Classic upgrades, the default is 2
- `-T` Takes offline user schema-based table spaces.
- `-z` Turns on production debugging information for `catcon.pm`.

Examples

```
global.catctl_options=-n 24 -N 4
```

Related Topics

- Upgrade Script (catctl.pl) Parameters

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.

Usage Notes

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.

Usage Notes

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.

Usage Notes

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
```

global_log_dir

(Optional for AutoUpgade Patching) Sets the location of the AutoUpgrade patching log files, and temporary files that belong to global modules, which AutoUpgrade uses.

Usage Notes

This configuration parameter is only used with AutoUpgrade patching. Its purpose is similar to the `autoupg_log_dir` parameter, which is not supported by AutoUpgrade Patching. You can configure a different log directory path in the `userconfig` file in the logs directory for a specific prefix.

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_patching`.

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.

Examples

```
global.global_log_dir=/path/to/my/global/log/dir
```

Configure different log directory path in the `userconfig` file in the logs directory for a specific prefix

```
global.global_log_dir=/path/to/my/global/log/dir
myprefix.log_dir=global.global_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
```

The following is an example of a configuration file suing the global_log_dir parameter:

```
global.global_log_dir=/logs/patching
global.keystore=/secure/keystore
upg1.sid=DB19X
upg1.source_home=/databases/19x/dbhome_1
upg1.target_home=/databases/19x/dbhome_2
upg1.folder=/storage/patches
upg1.download=YES
```

json_progress_writing_interval

(Optional) Sets the time interval for how often to write the AutoUpgrade progress JSON report.

Usage Notes

This parameter specifies how often the AutoUpgrade progress JSON report is written. If you do not set this parameter, then by default the AutoUpgrade progress JSON report interval is 30 seconds

Example

In the following example, `global.json_progress_writing_interval=90` is used to specify that the JSON progress report is written every 90 seconds to the log directory specified by `global.autoupg_log_dir`:

```
global.json_progress_writing_interval=90
global.autoupg_log_dir=/path/to/my/global/log/dir
```

keystore

(Optional) Specifies the location for a dedicated software keystore used exclusively by AutoUpgrade to store passwords, and other sensitive information.

Usage Notes

You can use the keystore parameter to specify where you want AutoUpgrade to create a dedicated software keystore that is used exclusively by AutoUpgrade.

The AutoUpgrade keystore contains the file `ewallet.p12` (similar to other kind of keystores used by the database). The file is created when you use the `save` command in the TDE prompt. If you choose to generate an auto-login keystore, then the file `cwallet.sso` is created as well. If you have an auto-login keystore, then AutoUpgrade does not prompt for a keystore password when AutoUpgrade starts.

The keystore generated by AutoUpgrade contains sensitive information, and is protected by a password that you choose when the keystore is used for the first time. Each time changes are made to the keystore, the password must be supplied. Unless you decide to create an auto-login keystore for AutoUpgrade, each time you start AutoUpgrade, and AutoUpgrade requires information from the keystore, you must provide the keystore password.

> **Caution:**
>
> Because the directory you specify with `global.keystore` contains a software keystore, it should be protected using the same security best practices as you use with all other highly secure keystore files.

Example

In the following example, replace *`ORACLE_SID`* with the system identifier of the database using the keystore.

```
global.keystore=/etc/oracle/keystores/ORACLE_SID/autoupgrade
```

raise_compatible

(Optional) Increases the compatible parameter to the default value of the target release after the upgrade is completed successfully.

Usage Notes

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`.

> **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`.

Options

`[yes | no]`

The default value is `no`.

Example

```
global.raise_compatible=yes
```

replay

(Optional) Specifies whether to use replay to upgrade the database.

Usage Notes

By default, AutoUpgrade performs a Classic upgrade to upgrade the database.

Options

`[yes | no]`

The default value is `no`.

Example

```
global.replay=yes
```

target_base

(Optional) Specifies the target `ORACLE_BASE` path for the target Oracle home.

Usage Notes

Use of this parameter is only required in rare cases.

Example

```
global.target_base=/u01/app/oracle
sales4.target_base=/u04/app/oracle4
```

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.

Usage Notes

Use this option to avoid specifying the same `target_home` multiple times. This parameter can be overwritten locally.

Earlier releases of AutoUpgrade required you to set `target_home` and `target_version` in the configuration file. In later releases of AutoUpgrade, this restriction has been lifted for both `Analyze` and `Fixups` modes. However, if you don't set `target_home`, then you must specify `target_version`. Either one of them must be present.

Example

```
global.target_home=/target/Oracle/home
```

target_version

(Optional) Specifies the target release version on which you want AutoUpgrade to perform the upgrade.

Usage Notes

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
- 23

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=19
employees.target_version=12.2
```

upgradexml

(Optional) Generates the `upgrade.xml` file.

Usage Notes

The `upgrade.xml` generated 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.

Options

`[yes | no]`

The default value is `no`.

Example

```
global.upgradexml=yes
```

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 Managed Configurations
  When AutoUpgrade detects Oracle RAC or Oracle RAC One Node, 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.
- Preparing for Oracle RAC Upgrades Using AutoUpgrade
  Review to find what information you must collect before the upgrade, and other upgrade preparation guidelines.

- AutoUpgrade and Oracle Data Guard
  Starting with Oracle Database 21c, AutoUpgrade can simplify the upgrade process for your primary and secondary databases configured for Oracle Data Guard.
- How to Run AutoUpgrade Using the Fast Deploy Option
  To minimize downtime, you can upgrade your database by running AutoUpgrade using the Fast Deploy option.
- How to Perform an Unplug-Plug Upgrade of an Encrypted PDB
  Learn how you can perform unplug-plug upgrades of encrypted PDBs using the AutoUpgrade Utility.
- How to Perform a Non-CDB to PDB Conversion of an Encrypted PDB
  With AutoUpgrade 22.1 and later updates, AutoUpgrade simplifies the upgrade and conversion of Oracle Databases that use Transparent Data Encryption (TDE).

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 you can recover from a failed 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 use the `target_pdb_copy_option` in your configuration file to create copies of your data files, then your existing database is available as a backup. This is a safe option, but will require additional time and disk space. 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 4-4 Upgrading and Converting a Non-CDB to Oracle Database 19c Using Multitenant Architecture

During the Deploy conversion and upgrade workflow, AutoUpgrade 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, and you did not choose to use the `target_pdb_copy_option` in your configuration file to create copies of your data files, then be aware that AutoUpgrade cannot recover and restore the database. In that event, you must restore the database manually.

AutoUpgrade Process Flow for Oracle Grid Infrastructure Managed Configurations

When AutoUpgrade detects Oracle RAC or Oracle RAC One Node, it proceeds to perform upgrade steps required for all Oracle RAC instances.

When you start AutoUpgrade, it detects when Oracle Database is managed with Oracle Grid Infrastructure, either with Oracle Real Application Clusters (Oracle RAC), or an Oracle RAC One Node configuration. Before you start AutoUpgrade, you must already have upgraded Oracle Grid Infrastructure to a release equal to or more recent than the Oracle Database release to which you are upgrading.

> **Note:**
>
> Choosing this upgrade option requires downtime of the clustered database while AutoUpgrade completes upgrades of database instances, and system configuration. If you use Oracle Enterprise Manager, then you must reconfigure it after the upgrade.

During an upgrade, when AutoUpgrade detects that the Oracle Database is an Oracle Clusterware resource, it performs the following steps, in sequence:

1. AutoUpgrade shuts down the database, or all instances of the Oracle RAC database.
2. AutoUpgrade disables Oracle RAC and Oracle RAC One Node services.
3. If the Oracle Clusterware resource is Oracle RAC, then AutoUpgrade disables the Oracle RAC database resource in Oracle Clusterware.
4. AutoUpgrade starts up the Oracle Database instance:
   - If the instance was an Oracle RAC instance, then AutoUpgrade starts the local Oracle Database instance in upgrade mode, with `CLUSTER_DATABASE` set to false.
   - If the instance was a single-instance Oracle Database, then it starts up the instance in normal mode.
5. AutoUpgrade upgrades the local Oracle Database Oracle home binaries to the new Oracle Database release binaries.
6. During an upgrade, AutoUpgrade runs `srvctl upgrade database` from the local Oracle Database home, and for Oracle RAC, upgrades the configuration of the Oracle RAC services to the new release. During a patch operation, instead of `srvctl upgrade database`, AutoUpgrade runs `srvctl modify database`.
7. AutoUpgrade enables Oracle Grid Infrastructure services for the database, using `srvctl enable database`. For Oracle RAC instances, it changes `CLUSTER_DATABASE` to true.

8. AutoUpgrade recreates the server parameter file (`SPFILE`) with the updated parameters, and the parameter options you previously set for your environment that are not affected by the release upgrade.
9. If the Oracle Database is an instance of an Oracle RAC database, then AutoUpgrade repeats this process on each other cluster member node, until all instances of the Oracle RAC database are upgraded.
10. AutoUpgrade starts up the Oracle Database. For Oracle RAC, it starts all instances of Oracle Real Application Clusters on the cluster.

During a patch operation, the process is similar, with the following differences:

1. AutoUpgrade shuts down the database, or all instances of the Oracle RAC database.
2. AutoUpgrade disables Oracle RAC and Oracle RAC One Node services.
3. If the Oracle Clusterware resource is Oracle RAC, then AutoUpgrade disables the Oracle RAC database resource in Oracle Clusterware.
4. AutoUpgrade starts up the Oracle Database instance:
   - If the instance was an Oracle RAC instance, then AutoUpgrade starts the local Oracle Database instance in upgrade mode, with `CLUSTER_DATABASE` set to false.
   - If the instance was a single-instance Oracle Database, then it starts up the instance in normal mode.

   In comparison to an upgrade operation, patch maintenance also performs the following steps in this stage, so that the database is ready to start up after the drain stage is complete:
   - AutoUpgrade upgrades the local Oracle Database Oracle home binaries to the new Oracle Database release binaries.
   - AutoUpgrade runs `srvctl modify database` from the local Oracle Database home, and patches the database. For Oracle RAC, it also patches the configuration of the Oracle RAC services to the new release update.
   - AutoUpgrade enables Oracle Grid Infrastructure services for the database, using `srvctl enable database`. F
   - AutoUpgrade recreates the server parameter file (`SPFILE`) with the updated parameters, and the parameter options you previously set for your environment that are not affected by the release patch maintenance.
   - If the Oracle Database is an instance of an Oracle RAC database, then AutoUpgrade repeats this process on each other cluster member node, until all instances of the Oracle RAC database are upgraded.
   - AutoUpgrade starts up the Oracle Database. For Oracle RAC, it starts all instances of Oracle Real Application Clusters on the cluster.

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:

- Using AutoUpgrade with Oracle RAC deployments is now support for both POSIX systems and Microsoft Windows systems.
- For Microsoft Windows only, AutoUpgrade has integrated the Oracle RAC API, so there is no requirement to have SSH enabled on the Windows platforms. AutoUpgrade is supported with releases 19, 21 and 23 of Oracle Grid Infrastructure on Microsoft Windows. Database upgrades and patching are supported for databases that run in the supported Oracle Grid Infrastructure releases. Oracle recommends using the AutoUpgrade Windows Credentials feature as a complete solution when upgrading or patching in a Microsoft Windows Oracle RAC environment.
- 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

Preparing for Oracle RAC Upgrades Using AutoUpgrade

Review to find what information you must collect before the upgrade, and other upgrade preparation guidelines.

To use AutoUpgrade for Oracle Real Application Clusters (Oracle RAC) upgrades, ensure that you collect information as needed before the upgrade, and be prepared to provide information during the upgrade.

Scope Limits for AutoUpgrade and Oracle RAC

AutoUpgrade does not perform upgrades of the Oracle Clusterware and Oracle ASM components of Oracle Grid Infrastructure. Before you start AutoUpgrade to upgrade your Oracle RAC database, you must first complete a successful Oracle Grid Infrastructure upgrade to the new release.

File System Preparation Before Upgrades Using AutoUpgrade

AutoUpgrade can identify the `PFILE` and `SPFILE` files shared on Oracle ASM. AutoUpgrade recreates the SPFILE as part of the upgrade. If you are sharing an `SPFILE` on the cluster using Oracle ASM, then you do not need to complete this procedure.

AutoUpgrade and Oracle Data Guard

Starting with Oracle Database 21c, AutoUpgrade 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 4-5 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 4-6 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

How to Run AutoUpgrade Using the Fast Deploy Option

To minimize downtime, you can upgrade your database by running AutoUpgrade using the Fast Deploy option.

Starting with AutoUpgrade 21.2, if your applications require minimal downtime, you can now upgrade with less downtime by using Fast Deploy. With the Fast Deploy option, you can run the prechecks and prefixups while the database is still online. After the fixups run on the source database, you can then run AutoUpgrade in Deploy mode, and skip the prechecks and prefixups stages, so that only the actual upgrade requires downtime.

> **Note:**
>
> Oracle recommends that you run AutoUpgrade using standard Analyze and Deploy modes. If you choose to use the Fast Deploy method, then be aware that there is a small risk that new issues can develop in the time duration after you run AutoUpgrade in the preupgrade Analyze mode and before you run AutoUpgrade in Upgrade mode, which can cause a problem. Assess that risk, and take precautions accordingly.

1. Create your AutoUpgrade configuration file, providing information about your source and target systems, and your upgrade preferences. In the steps that follow, that file name is `myconfig.cfg`.
2. Analyze the database using Analyze mode.

   ```
   - java -jar autoupgrade.jar -config myconfig.cfg -mode analyze
   ```

3. Run the preupgrade fixups using Fixups mode.

   ```
   - java -jar autoupgrade.jar -config myconfig.cfg -mode fixups
   ```

4. Upgrade the database using Upgrade mode.

   ```
   - java -jar autoupgrade.jar -config myconfig.cfg -mode upgrade
   ```

   As this command runs, the database can experience downtime, because databases being upgraded are opened in `UPGRADE` mode in the target Oracle home.

How to Perform an Unplug-Plug Upgrade of an Encrypted PDB

Learn how you can perform unplug-plug upgrades of encrypted PDBs using the AutoUpgrade Utility.

> **Caution:**
>
> If you choose to specify the directory for AutoUpgrade to create with `global.keystore`, then be aware that it contains a software keystore. It should be protected using the same security best practices as you use with the TDE keystore files.

To perform upgrades of encrypted PDBs, AutoUpgrade requires loading the password for the TDE keystore into its own secure keystore. To load the passwords, you set the AutoUpgrade global configuration file parameter `keystore` in your configuration file, and run AutoUpgrade using the command-line parameter `load_password`. This parameter must be used in conjunction with the `-config` parameter. It takes no arguments. Instead, it starts an interactive prompt with specific commands that enable you to provide information required for the keystore. AutoUpgrade stores the passwords you load securely during the upgrade, and uses those passwords when needed.

> **Note:**
>
> If the database is using an Oracle Secure External Password Store (SEPS), and a TDE keystore password is required, then AutoUpgrade uses the `IDENTIFIED BY EXTERNAL STORE` clause, so it does not require loading passwords into the AutoUpgrade password keystore. However, if all databases are configured with a Secure External Password Store, then you still need to define `global.keystore` in your configuration file.
>
> The AutoUpgrade keystore is similar to other keystores that Oracle Database uses. You have the option to create it as an auto-login keystore. For example, if the external keystore is `ewallet.p12` AutoUpgrade creates an auto-login keystore `cwallet.sso`, which is used to open the `ewallet.p12` keystore.

Before you begin to upgrade the encrypted PDB, the following must be true:

- AutoUpgrade must be version 22.2 or later. Oracle strongly recommends that you always download and run the latest version of AutoUpgrade.
- You must use an external TDE keystore
- You have to have available to you the external keystore password of the source and target CDB.

Example 4-7 Upgrading an Encrypted PDB with an Unplug Plug Upgrade Using AutoUpgrade

In the following example, an Oracle Database 12c Release 2 (12.2) PDB using Transparent Data Encryption (TDE) is upgraded to Oracle Database 19c, using the AutoUpgrade `load_password` command option with the AutoUpgrade configuration file keystore parameter to provide a secure store for the TDE keys during the upgrade.

1. Ensure that you have the latest version of AutoUpgrade:

```
$ java -jar autoupgrade.jar -version
```

   If the AutoUpgrade version is an earlier release, then download the most recent version of AutoUpgrade from My Oracle Support AutoUpgrade Tool (Doc ID 2485457.1)

2. Create your AutoUpgrade configuration file. In this example, the configuration file `PDB1.cfg` is created with the path to the keystore, which is specified in the `admin` folder under the Oracle base directory, in the path `/u01/app/oracle/admin/autoupgrade/keystore`. The source CDB is `CDB1`,and the target CDB is `CDB2`:

```
global.autoupg_log_dir=/u01/app/oracle/cfgtoollogs/autoupgrade
global.keystore=/u01/app/oracle/admin/autoupgrade/keystore
upg1.log_dir=/u01/app/oracle/cfgtoollogs/autoupgrade/DB12
upg1.source_home=/u01/app/oracle/product/12.2.0/dbhome_1
upg1.target_home=/u01/app/oracle/product/19.1.0/dbhome_1
upg1.sid=CDB1
upg.pdbs=PDB1
upg1.target_cdb=CDB2
```

   > **Note:**
   >
   > If you do not specify a keystore location, and an AutoUpgrade keystore has not been created previously, then AutoUpgrade creates the AutoUpgrade keystore for you.

3. Run AutoUpgrade in Analyze mode:

```
$ java -jar autoupgrade.jar -config PDB1.cfg -mode analyze
```

   The summary report indicates that TDE keystore passwords are needed before starting the upgrade (`TDE_PASSWORDS_REQURED`):

```
[Stage Name]    PRECHECKS
[Status]        FAILURE
[Start Time]    2022-03-29 07:58:52
[Duration]
[Log Directory] /u01/app/oracle/cfgtoollogs/autoupgrade/PDB1/CDB1/100/
prechecks
[Detail]        /u01/app/oracle/cfgtoollogs/autoupgrade/PDB1/CDB1/100/
prechecks/cdb1_preupgrade.log
        Check failed for PDB1, manual intervention needed for the below
checks
        [TDE_PASSWORDS_REQUIRED]
```

4. Add the TDE keystore passwords into the AutoUpgrade keystore:

```
$ java -jar autoupgrade.jar -config PDB1.cfg -load_password
```

The first time you run AutoUpgrade with the `-load_password` command option, you are prompted to create a password for the AutoUpgrade keystore, where TDE passwords can be stored securely:

```
Starting AutoUpgrade Password Loader - Type help for available options
Creating new keystore - Password required
Enter password:
Enter password again:
Keystore was successfully created
```

If do not use a SEPS keystore, then AutoUpgrade prompts you to add the TDE keystore passwords for the databases specified with `source_home` in your configuration file to AutoUpgrade's own keystore, where that database requires a TDE keystore password. This is what we have specified in the configuration file example.

In the following example, the source and target TDE keystore passwords are loaded:

```
TDE> ADD CDB1
Enter your secret/Password:
Re-enter your secret/Password:

TDE> ADD CDB2
Enter your secret/Password:
Re-enter your secret/Password:
```

Confirm that the passwords are loaded:

```
TDE> LIST
+--------------+----------------+----------------+-------------
+----------------------+
|ORACLE_SID     |Action Required  |TDE Password     |SEPS Status   |Active
Wallet Type    |
+--------------+----------------+----------------+-------------
+----------------------+
| CDB1          |                 | Verified        | Inactive     | Auto-
login          |
| CDB2          |                 | Verified        | Inactive     |
Any                  |
+--------------+----------------+----------------+-------------
+----------------------+
```

When the `Action Required` column is empty, the TDE passwords are available for the upgrade.

If you use a SEPS keystore on the source CDB or target CDB, then AutoUpgrade automatically detects the SEPS keystore as the source for the TDE password. AutoUpgrade uses the `IDENTIFIED BY EXTERNAL STORE` clause, and does not need to load the TDE keystore passwords to the AutoUpgrade keystore. Again, the TDE LIST command should show an empty `Action Required` column:

```
TDE> LIST
+-------------+----------------+-------------------------+-------------
+---------------------+
|ORACLE_SID    |Action Required    | TDE Password           |SEPS Status
```

```
|Active Wallet Type   |
+-------------+-------------------+-----------------------+-------------
+--------------------+
| CDB1        |                   |No password loaded     | Verified
| Any                |
| CDB2        |                   |No password loaded     | Verified
| Any                |
+-------------+-------------------+-----------------------+-------------
+--------------------+
```

Both options can be used in the same configuration file. For example, if you do not use a SEPS keystore for the source non-CDB database, but you do use a SEPS keystore for the target CDB, then you only need to load a password for the source non-CDB.

5. Save the TDE passwords into the AutoUpgrade keystore. (Optional): In this example, we will convert the keystore to an auto-login keystore:

```
TDE> save
Convert the keystore to auto-login [YES|NO] ? YES
TDE> exit
```

6. Analyze the PDB again:

```
$ java -jar autoupgrade.jar -config PDB1.cfg -mode analyze
```

Review the report and confirm all issues are resolved.

7. Start the upgrade and conversion:

```
$ java -jar autoupgrade.jar -config PDB1.cfg -mode deploy
```

8. AutoUpgrade proceeds to upgrade `PDB1` on the target Oracle Database, using the TDE passwords as needed to complete the upgrade. Because in this example we have configured AutoUpgrade to create an AutoUpgrade auto-login keystore, and access the TDE passwords using its own secure keystore, you are not prompted to provide TDE passwords during the upgrade.

Related Topics

- Managing the Secure External Password Store for Password Credentials
- Upgrading an Encrypted PDB
- AutoUpgrade Tool (Doc ID 2485457.1)

How to Perform a Non-CDB to PDB Conversion of an Encrypted PDB

With AutoUpgrade 22.1 and later updates, AutoUpgrade simplifies the upgrade and conversion of Oracle Databases that use Transparent Data Encryption (TDE).

To protect sensitive information during upgrades while simplifying the upgrade process, you can use the AutoUpgrade global configuration file parameter `keystore`, and the AutoUpgrade command-line parameter `load_password` to load TDE passwords securely into the AutoUpgrade keystore, and use those passwords when needed.

Before you can use the AutoUpgrade keystore, you specify the location of the external password store in your AutoUpgrade configuration file using `global.keystore`.

> **Note:**
>
> If you specify the AutoUpgrade keystore path, then that path should be different from any other file path you specify in AutoUpgrade, so that the keystore is not in any log file location. If the AutoUpgrade keystore path directory does not exist, then AutoUpgrade automatically creates it..
>
> If the database is using an Oracle Secure External Password Store (SEPS), and a TDE keystore password is required, then AutoUpgrade uses the `IDENTIFIED BY EXTERNAL STORE` clause, so it does not require loading passwords into the AutoUpgrade password keystore. However, if all databases are configured with a Secure External Password Store, you still need to define `global.keystore` in your configuration file.

Example 4-8 Upgrading a Database Using TDE and Converting from Non-CDB to PDB

In the following example, an Oracle Database 12c Release 2 (12.2) non-CDB database using Transparent Data Encryption (TDE) is upgraded to Oracle Database 19c and converted to a PDB, using the AutoUpgrade `load_password` command option with the AutoUpgrade configuration file keystore parameter to provide a secure store for the TDE keys during PDB conversion and upgrade.

1. Ensure that you have the latest version of AutoUpgrade:

   ```
   $ java -jar autoupgrade.jar -version
   ```

   If the AutoUpgrade version is an earlier release, then download the most recent version of AutoUpgrade from My Oracle Support AutoUpgrade Tool (Doc ID 2485457.1)

2. Create your AutoUpgrade configuration file. In this example, the path to the keystore is specified in the admin folder under the Oracle base directory, in the path `/u01/app/oracle/admin/autoupgrade/keystore`:

   ```
   global.autoupg_log_dir=/u01/app/oracle/cfgtoollogs/autoupgrade
   global.keystore=/u01/app/oracle/admin/autoupgrade/keystore
   upg1.log_dir=/u01/app/oracle/cfgtoollogs/autoupgrade/DB12
   upg1.source_home=/u01/app/oracle/product/12.2.0
   upg1.target_home=/u01/app/oracle/product/19.1.0
   upg1.sid=DB12
   upg1.target_cdb=CDB2
   ```

3. Run AutoUpgrade in Analyze mode:

   ```
   $ java -jar autoupgrade.jar -config DB12.cfg -mode analyze
   ```

   The summary report indicates that no additional preupgrade tasks need to be completed before starting the upgrade:

   ```
   [Stage Name]    PRECHECKS
   [Status]        SUCCESS
   [Start Time]    2022-03-30 10:28:38
   [Duration]
   [Log Directory] /u01/app/oracle/cfgtoollogs/autoupgrade/DB12/DB12/100/
   ```

```
prechecks
[Detail]        /u01/app/oracle/cfgtoollogs/autoupgrade/DB12/DB12/100/
prechecks/db12_preupgrade.log
                Check passed and no manual intervention needed
```

If the `TDE_PASSWORDS_REQUIRED` check fails, then load the TDE password:

```
$ java -jar autoupgrade.jar -config DB12.cfg -load_password
```

The first time you run AutoUpgrade with the `-load_password` command option, if AutoUpgrade requires a TDE password to perform the upgrade, then you are prompted to create a password for the AutoUpgrade keystore, where the TDE password can be stored securely:

```
Starting AutoUpgrade Password Loader - Type help for available options
Creating new keystore - Password required
Enter password:
Enter password again:
Keystore was successfully created
```

If do not use a SEPS keystore, then AutoUpgrade prompts you to add the TDE keystore passwords for the databases specified with `source_home` in your configuration file to AutoUpgrade's own keystore, where that database requires a TDE keystore password. This is what we have specified in the configuration file example.

In the following example, the source and target TDE keystore passwords are loaded:

```
TDE> ADD DB12
Enter your secret/Password:
Re-enter your secret/Password:

TDE> ADD CDB2
Enter your secret/Password:
Re-enter your secret/Password:
```

4. To check the TDE configuration, you can run AutoUpgrade again using the `-load_password` command parameter. This time, because the password is already loaded, the TDE console prompt appears. Run the TDE console `list` command to check the TDE configuration:

```
$ java -jar autoupgrade.jar -config DB12.cfg -load_password

TDE> list
+----------+---------------+-----------------+-----------
+------------------+
|ORACLE_SID|Action Required|   TDE Password   |SEPS Status|Active Wallet
Type|
+----------+---------------+-----------------+-----------
+------------------+
|      CDB2|               |No password loaded|   Verified|
Any|
|      DB12|               |No password loaded|    Unknown|        Auto-
login|
```

```
+---------+--------------+-----------------+----------
+-----------------+
```

In the table output for the list command, the `Action Required` column is empty. This result verifies that you have provided the TDE keystore password. In the `SEPS Status` column, AutoUpgrade reports that it checked the password storage access on the target Oracle Database, `CDB2`, and confirms that the password works. Because the TDE console functionality to check the password was a feature added in Oracle Database 19c, AutoUpgrade is unable to confirm the password check for TDE on Oracle Database 12c Release 2 (12.2), so the result `Unknown` is expected.

If you use a SEPS keystore on the source CDB or target CDB, then AutoUpgrade automatically detects the SEPS keystore as the source for the TDE password. AutoUpgrade uses the `IDENTIFIED BY EXTERNAL STORE` clause, and does not need to load the TDE keystore passwords to the AutoUpgrade keystore. Again, the TDE LIST command should show an empty `Action Required` column.

Both options can be used in the same configuration file. For example, if you do not use a SEPS keystore for the source non-CDB database, but you do use a SEPS keystore for the target CDB, then you only need to load a password for the source non-CDB.

5. Start the upgrade and conversion:

```
$ java -jar autoupgrade.jar -config DB12.cfg -mode deploy
```

6. AutoUpgrade proceeds to upgrade and convert the source non-CDB Oracle Database to a PDB on the target Oracle Database, using the TDE passwords as needed to complete the upgrade.

> **Caution:**
>
> As with any other Oracle Database keystore, protect the AutoUpgrade keystore files:
>
> - Apply restrictive file system permissions
> - Audit access
> - Back up the keystore

Related Topics

- Managing the Secure External Password Store for Password Credentials
- AutoUpgrade and Secure External Password Store Enables Complete Automation
- AutoUpgrade Tool (Doc ID 2485457.1)

AutoUpgrade Patching

AutoUpgrade has enhanced capability that enables it to automate all of the necessary steps in the database patching process, including optional downloading of patches.

- How AutoUpgrade Performs AutoUpgrade Patching
  AutoUpgrade Patching extends the AutoUpgrade upgrade process to patching, which enables you to perform out-of-place patching for multiple databases using a single command.

- AutoUpgrade Patching Stages and Workflows
  AutoUpgrade Patching supports three different job modes. Each mode runs a different set of stages.
- PDB Priority and AutoUpgrade
  For both AutoUpgrade patching and upgrades, AutoUpgrade honors how PDBs will be prioritized by setting the priority on specifuc PDBs.
- Configuration Parameters and Command-Line Options for AutoUprade Patching
  In addition to other configuration file parameters and command-line options, there are specific parameters and options for AutoUpgrade Patching.
- AutoUpgrade Patching Configuration Files and Log Files
  See examples of configuration files and log files for AutoUpgrade Patching.
- AutoUpgrade Patching Workflow Examples
  Use this example of an AutoUpgrade patchiing workflow to understand the AutoUpgrade patching procedure.

How AutoUpgrade Performs AutoUpgrade Patching

AutoUpgrade Patching extends the AutoUpgrade upgrade process to patching, which enables you to perform out-of-place patching for multiple databases using a single command.

> **Note:**
>
> Oracle recommends that you use Oracle Fleet Patching and Provisioning for Oracle Real Application Clusters environments.

With the latest release of AutoUpgrade the AutoUpgrade Patching procedure can be performed on Release Update (RU), Monthly Recommended Patches (MRPs), and one-off patches, using out-of-place patching. When you patch from an earlier RU or MRP with AutoUpgrade, the simplicity, reliability, and recoverability of AutoUpgrade is extended to the patching process. As a result, patching is easier to perform, and you also obtain simpler recovery from any issues that can arise during the patch deployment.

There are no additional parameters or options that you need to set in the configuration file for AutoUpgrade to perform patching. Just specify the source and target Oracle homes, and AutoUpgrade will apply changes to the databases. When the source and target Oracle homes are the same Oracle Database release (for example, from 19.11 to 19.13), AutoUpgrade identifies the operation as an RU or MRP patch operation. This capability also applies to one-off patches: If the RU or MRP applied to the target Oracle home is later than the RU or MRP of the source Oracle home, then you can apply one-off patches as needed to the target Oracle home using this method.

As AutoUpgrade performs the patch operation, it uses Datapatch to apply an RU or MRP to databases. The process of patching takes advantage of existing AutoUpgrade options and operations, as with a full release upgrade, including creating a Guarantee Restore Point. During the patching process, the database is shut down in the source Oracle home, and brought back up in the target Oracle home. RU or MRP updates are performed in upgrade mode, using Datapatch.

> **Note:**
>
> AutoUpgrade does not perform rolling upgrades of Oracle Database releases. Oracle recommends that you use Oracle Fleet Patching and Provisioning to be able to obtain rolling patch maintenance for Oracle Real Application Clusters (Oracle RAC) environments.

Benefits of AutoUpgrade Patching

- AutoUpgrade enables patching of all databases specified in the configuration file, in one operation.
- Resume capabilities are already included with AutoUpgrade.
- Restore capabilities are provided so that it is possible to roll back to the earlier release Oracle home:
  - Restore using the Guarantee Restore Point (GRP) generated during the patching process.
  - If a GRP is not present, then the Datapatch rollback functionality can also perform a restore. AutoUpgrade detects that a GRP does not exist, and automatically performs the restore using Datapatch. To enable a Datapatch rollback restore, set the restoration configuration option to `no`. For example: `sales1.restoration=no`.
- Oracle RAC management is provided automatically with AutoUpgrade.
- The error management reporting features of AutoUpgrade are extended to patching.
- The JSON status and progress reporting of AutoUpgrade are extended to patching.
- AutoUpgrade uses the Datapatch JSON status files to determine success or failure of each process, and reports those results to you at the completion of the process.

Features Supported for AutoUpgrade Patching

The following features are provided with AutoUpgrade Patching

- Patching of databases encrypted with Transparent Data Encryption (TDE).
- Hot Cloning/Relocate, which enables you to create PDBs from a Non-CDB, or from a PDB on a remote host.
- Proactive Fixups, which runs postfixups on the PDB as soon as the PDB is patched.
- Multi-node patching and upgrades using Oracle Real Application Clusters (Oracle RAC), which enables AutoUpgrade to spread the workload across multiple nodes.
- Configuration management, by copying or merging the source Oracle home configuration files (`tnsnames.ora`, `sqlnet.ora`, and other files) to the target Oracle home.
- Analysis of the database before starting a patching procedure, to ensure patch readiness.
- Running fixups during production before patching, so that you reduce downtime. You can then use `-mode upgrade` to bypass running fixups and proceed directly to patching the database.
- Performing all patching tasks in Deploy mode, to achieve end-to-end patching.
- Enhanced reporting of the patching process, which enables you to diagnose errors more easily by using the `datapatch_summary.log` report.
- Datapatch log files found in the Datapatch summary JSON file are copied to the output file of the apply or rollback operational logs as follows:

  - Apply log format, where *dbname* is the name of the database: `applydatapatchlogfiles#dbname.log`
  - Rollback log format, where *dbname* is the name of the database: `rollbackdatapatchlogfiles#dbname.log`
- Datapatch output is logged to following operational logs:
  - Apply log format, where *dbname* is the name of the database: `applyautoupgrade#dbname.log`
  - Rollback log format, where *dbname* is the name of the database: `rollbackautoupgrade#dbname.log`
- Datapatch JSON output is logged to the following operational logs:
  - Apply log format, where *dbname* is the name of the database: `applydatapatchsummary#dbname.log`
  - Rollback log format, where *dbname* is the name of the database: `rollbackdatapatchsummary#dbname.log`
- Storage of all related patching log files produced by AutoUpgrade are stored in the `dbupgrade` directory.
- Standard Datapatch log files are stored in *Oracle-base*`/cfgtoollogs/sqlpatch`.
- Starting with AutoUpgrade 23.1, Patching is supported for single-instance databases on Microsoft Windows. It is not supported with Oracle Real Application Clusters (Oracle RAC).

Requirements and Restrictions for using AutoUpgrade Patching

- Downtime is required to use AutoUpgrade Patching:
  - After the patch operation is completed, the database is restarted.
  - Patching in normal mode enables the database to be available in the target home while the patching is proceeding.
  - Patching in upgrade mode requires the patching to be completed before the database is restarted in the target home.
  - During the patch operation, the CDB and PDBs are opened in normal mode after drain operations complete, and patching proceeds. However, rolling patching for Oracle Real Application Clusters is not supported at this time.
  - To enable PDBs to become available as soon as they are successfully patched, then ensure that you set the `tune_settings` option `make_pdbs_avaliable` to `true` in your configuration file. For example: `sales3.tune_setting=proactive_fixups=true,make_pdbs_available=true`
  - Because downtime is required, rolling upgrades of Oracle RAC are not supported at this time.
- AutoUpgrade can support a target RU or MRP that has additional patches applied. However, the source Oracle Database must be on an earlier RU or MRP than the target Oracle Database RU or MRP.
- AutoUpgrade does not move the listener to the new Oracle home. If needed, you must manually move the listener to the new Oracle home before you start AutoUpgrade.
- Installation and configuration of the target Oracle Database home must be complete before you can run RUs and MRPs on the target home using AutoUpgrade Patching. If a target RU is patched after it has been applied to the database, then you can no longer use AutoUpgrade Patching. Instead, you must use Datapatch to apply the updates.

- Performing rolling patches of Oracle Database (single-instance or Oracle RAC) is not supported. When you use AutoUpgrade Patching on a single-instance or Oracle RAC Oracle Database, AutoUpgrade's Oracle RAC management will shut down the database on all nodes.
- The target Oracle Database RU must be at least Oracle Database 19c, RU 19.3, or later releases. You cannot use AutoUpgrade Patching to perform RUs or MRPs to earlier Oracle Database releases.
- The effect of AutoUpgrade Patching on databases using Oracle Data Guard Physical Standby or Oracle Data Guard Logical Standby and Rolling Upgrade is the same as with using Datapatch. AutoUpgrade patching can be a part of the procedures.

> **Note:**
>
> Oracle recommends that you use Oracle Fleet Patching and Provisioning for databases deployed with Oracle Data Guard

- AutoUpgrade supports physically moving a database to different hardware. When the database is moved to a different target system, AutoUpgrade Patching can be used with the `-mode upgrade` option. However, when the database is moved to a different system, AutoUpgrade is unable to perform a restore of the database. In that case, the upgrade options for patching follow the same rules that apply for upgrading your database.
- In describing AutoUpgrade, this documentation generally refers to AutoUpgrade performing upgrades. When AutoUpgrade performs patching, patching is functionally similar to upgrades. Accordingly, references to upgrades in this guide or in diagrams are applicable to both upgrade and patching with AutoUpgrade Patching.
- Performing patching with AutoUpgrade Patching supports all AutoUpgrade `-mode` options except the `preupgrade` mode, which is only supported for upgrades.
- When patching or upgrading relocatable PDBs the configuration file cannot contain a mixture of patch clone PDBs and upgraded clone PDBs. The configuration for the clone PDBs must be either all upgrade clones, or all patched clones.
- AutoUpgrade patching supports only one `catctl_options` setting, `-n` *`number`*, where *`number`* is the number of processes to use for parallel operations. The `catctl_options=-n` setting enables you to control the total number of PDBs that you want to run concurrently during the patching process. The default is `CPU_COUNT` divided by 2. For example if `CPU_COUNT` is set to 24 then by default, 12 PDBs (Datapatch instances) can run concurrently during the patching process.

  To overwrite the default, add *`prefix`*`.catctl_options` to your configuration file with a value for the number of concurrent Datapatch instances you want to run. For example, to configure AutoUpgrade to run 6 PDBs (Datapatch instances) for the patch operations specified with the prefix `sales`, overriding the default, add the following line to your configuration file:

  ```
  sales.catctl_options=-n 6
  ```

> **Caution:**
>
> For 19.13 and earlier release updates (RUs), Oracle strongly recommends that you set the `catctl_options` parameter for patching operations.
>
> With 19.13 and earlier RUs, the default value of SQL*Plus processors allocated to each Datapatch instance is `CPU_COUNT` multiplied by two (CPU_COUNT*2) processors for each Datapatch instance that AutoUpgrade starts. This default SQL*Plus processor allocation can quickly overload your system. To restrict system resources allocated to the patch operation, restricting how many Datapatch instances running simultaneously is the only option. For RU 19.13 and later, only one SQL*Plus process is started for each instance of Datapatch. This change enables AutoUpgrade to run more PDBs in parallel without consuming a large amount of system resources.

Security Characteristics of AutoUpgrade Patching

- The user running AutoUpgrade Patching must have the `SYSDBA` system privilege to log into the database, and run the patching operation. For an Oracle Real Application Clusters database, the user must be the owner of the source and target Oracle Database Oracle homes.
- The same security rules that apply for upgrades also apply with AutoUpgrade Patching.

Performance Characteristics of AutoUpgrade Patching

- The speed of deploying an RU or MRP depends on the number of PDBs in a CDB, and on changes in the release update or release update revision. If the number of changes from the source RU or MRP patches are relatively few, then deployment of patches should be quick. If the patches have many changes, then applying the patches requires more time.
- Because AutoUpgrade Patching includes a number of additional automated procedures, release update deployment is slightly slower than running Datapatch manually. For example, AutoUpgrade Patching automatically includes recompiling invalid objects, and configuring Oracle RAC management on the target system.

What Happens If AutoUpgrade Patching Rolls Back a Release Update

- AutoUpgrade uses the Guarantee Restore Point (GRP) generated during the patch process to roll back to the earlier release Oracle home.
- If no GRP is created, then AutoUpgrade automatically calls Datapatch to rollback the changes.

Related Topics

- Mike Dietrich Upgrade Your Database Now: AutoUpgrade's Patching: the feature you waited for

AutoUpgrade Patching Stages and Workflows

AutoUpgrade Patching supports three different job modes. Each mode runs a different set of stages.

AutoUpgrade Patching supports four different job modes. Each mode runs a different set of stages. In AutoUpgrade Patching operations, there are four job modes, with AutoUpgrade stages that are completed in these modes:

- **Analyze**: In this mode, the PRECHECKS stage is run.
- **Fixups**: In this mode, the PRECHECKS and PREFIXUPS stages are run.
- **Deploy**: In this mode, the following stages are run: GRP, PREACTIONS, PRECHECKS, PREFIXUPS, EXTRACT, DBTOOLS, INSTALL, ROOTSH, OPTIONS, OPATCH, AUTOUPGRADE, POSTCHECKS, POSTFIXUPS, POSTACTIONS, and DROPGRP.
- **create_home**: In this mode, the following stages are run: EXTRACT, INSTALL, ROOTSH, DBTOOLS, and OPATCH.

The stages performed during AutoUpgrade Patching are consistent with stages performed in other AutoUpgrade operations,but the descriptions that follow provide you with further information in a patching context.

Processes Performed During AutoUpgrade Patching Stages

- **GRP** When `restoration=YES` is set in the AutoUpgrade configuration file, creates a Guaranteed Restore Point (GRP) in the database before applying the patches
- **PREACTIONS** Runs any actions defined by the `before_action` parameter in the AutoUpgrade configuration file
- **PRECHECKS** Runs the patching prechecks
- **PREFIXUPS** Runs the fixups for any failing prechecks that have an automated fixup
- **EXTRACT** Extracts the base image that will be used to create the new target Oracle home.
- **DBTOOLS** Installs a newer version of OPatch into the target Oracle home, if necessary
- **INSTALL** Installs the new target Oracle home by running `runInstaller` in silent mode
- **ROOTSH** Reminds the user performing AutoUpgrade patching to run `root.sh` from the target Oracle home after patching is complete. This action is not automatically performed by AutoUpgrade Patching
- **OPTIONS** Synchronizes the binary options from the source `ORACLE_HOME` with the newly created target `ORACLE_HOME`. The supported options include the following:
  - `ASM` (Oracle Automatic Storage Management)
  - `OLAP` (Online Analytic Processing)
  - `PART` (Partitioning)
  - `RAT` (Oracle Real Application Testing)
  - `UNIAUD` (Unified Auditing)
- **OPATCH** Installs each of the necessary patches
- **PATCHING** Runs a second copy of AutoUpgrade without the `-patch` option to move the database from the source Oracle home to the target Oracle home
- **POSTCHECKS** Runs the patching postchecks
- **POSTFIXUPS** Runs the fixups for any failing postchecks that have an automated fixup
- **POSTACTIONS** Runs any actions defined by the `after_action` parameter in the AutoUpgrade configuration file
- **DROPGRP** Drops the guaranteed restore point (GRP) if one was created by AutoUpgrade Patching, and if `drop_grp_after_patching=YES` is set in the AutoUpgrade configuration file.

AutoUpgrade Patching Status and Progress Files

Status and progress files are available in the directory that you specify in the AutoUpgrade configuration file with the configuration parameter `global.global_log_dir`. The locattion of these files are placed as follows, where *`global-log`* is the directory path specified by `global.global_log_dir`:

`global_log``/cfgtoollogs/patch/auto/status/`

AutoUpgrade with the -patch parameter

> **Note:**
>
> Oracle recommends that you use Oracle Fleet Patching and Provisioning (FPP) for Oracle Real Application Clusters and deployments with Oracle Data Guard.

AutoUpgrade patching using the `-patch` parameter performs out-of-place patch maintenance, which is Oracle's recommended method of patching. In out-of-place patching, AutoUpgrade creates a new target Oracle home using a base image of the database's initial release. It installs the patches that you specify, and then moves the source database to the new target Oracle home. In addition, you can use AutoUpgrade for patch maintenance for database releases where you are unable to use FPP or DBCA

To use AutoUpgrade patching, your system should be consistent with the following requirements:

- Single-Instance Oracle Database
- Oracle Release 19c (19.3) or later release

PDB Priority and AutoUpgrade

For both AutoUpgrade patching and upgrades, AutoUpgrade honors how PDBs will be prioritized by setting the priority on specifuc PDBs.

The priority set on each of the PDBs is used to determine which PDBs will be patched or upgraded first. For both patching and upgrades, the control structure `CDB$ROOT` and the template structure `PDB$SEED` will always be updated first. All other PDBs will be updated after CDB$ROOT is updated. PDB$SEED will always be processed in the first batch. When priority is set on PDBs, the PDBs are updated in priority order. For example, PDBs with priority 3 will be processed before PDBs with priority 4. PDBs with no priority will be processed last.

This AutoUpgrade feature is supported for Oracle Database 12c release 2 and later releases.

Related Topics

- To Set the Priority of a PDB

Configuration Parameters and Command-Line Options for AutoUprade Patching

In addition to other configuration file parameters and command-line options, there are specific parameters and options for AutoUpgrade Patching.

To specify the AutoUpgrade Patching process options, you use configuration parameters that apply specifically patching.

> **Note:**
>
> AutoUpgrade Patching requires the Oracle Database 19.3 base image to be available in the directory specified by the `folder` configuration parameter. The base image can not be automatically downloaded by AutoUpgrade Patching, but can be done manually from this link:
>
> https://www.oracle.com/database/technologies/oracle19c-linux-downloads.html
>
> Downloading the Oracle Database 19.3 base image is a one-time operation. The same base image can be reused for all outofplace patching options that are performed by AutoUpgrade Patching.

Table 4-1 AutoUpgrade Patching Configuration File Parameters

Parameter	Description
`folder`	Specifies the directory that contains the patch zip files as well as the required base image. There is no default value. You must provide the directory path. When `download=YES`, this is the directory to which the patches will be downloaded When `download=NO`, this is the directory that must contain the patches that have been manually downloaded. The directory that you specify with the `folder` parameter must contain the base image of the source database's release (for example, Oracle Database Release 19.3).

`patch`	A comma-delimited list of the patches that you want to install.
	Note: When `method=outofplace` is set, then the patch list must include `RU,OPATCH`
	The default value is `RECOMMENDED`.
	Supported value:
	`RECOMMENDED`: Alias for all of the following options: `RU`, `OPATCH`, `OJVM`, `DPBP`.
	`RU`: Latest release update
	`RU:x.y`: A release update (RU) of the specified release version, where *x* is the major release number, and *y* is the RU. For example: `RU:19:24`
	`OPATCH`: Use latest version of OPatch
	`OJVM`: Apply the Oracle Java VM patch that applies to the specified RU.
	`OJVM:x.y`: Apply the Oracle Java VM patch of the specified release version, where *x* is the major release number, and *y* is the RU. For example: `RU:19.24`
	`DPBP`: Apply the Oracle Data Pump patch for the specified RU
	patch-number: Provide a specific one-off patch number. For example: `upg1.patch=p12345678_19.21.0.0.123456`

Table 4-1 (Cont.) AutoUpgrade Patching Configuration File Parameters

Parameter	Description
download	Specifies whether to automatically download patches from My Oracle Support. The default is YES. When set to YES, you must also load the My Oracle Support (MOS) credentials into AutoUpgrade Patching using the -load_password command-line option. The patches that are downloaded are placed into the directory folder specified by the folder parameter. If proxy information is required to connect to My Oracle Support, then set proxy values using the Linux operating system environment variables https_proxy, http_proxy, and no_proxy.
method	Specifies whether to create a new target Oracle home, and if so, how it will be created. The default value is OUTOFPLACE. OUTOFPLACE: Creates a new target Oracle home using the base image contained in the directory specified by the folder parameter.

For AutoUpgrade Patching, you must specify the following command-line options:

Table 4-2 AutoUpgrade Patching Command-Line Options

Command-Line Option	Description
target_home	Specifies the directory that will be used to create the new target Oracle home. This directory will be created by AutoUpgrade Patching and should not exist or be empty. Ensure that the location you specify on the filesystem contains sufficient free disk space for the creation of a new Oracle home.
sid	Specifies the ORACLE_SID value of the source database. If you specify -config_values and the ORACLE_SID Linux environment variable is defined, then this parameter becomes optional.
source_home	Specifies the Oracle home value of the source database. If you specify -config_values and the ORACLE_HOME Linux environment variable is defined, then this parameter becomes optional.
keystore	Specifies the secure directory (keystore) that will contain the keystore files. If you specify DOWNLOAD=NO, then this parameter becomes optional.

AutoUpgrade Patching Configuration Files and Log Files

See examples of configuration files and log files for AutoUpgrade Patching.

An AutoUpgrade Patching configuration file is essentially the same as an AutoUpgrade configuration file for upgrades. However, instead of AutoUpgrade using the parameters you specify to perform an upgrade, AutoUpgrade performs a patch operation from a source Oracle Database patch release to a target Oracle Database patch release, and AutoUpgrade runs Datapatch as part of the procedure.

Example 4-9 AutoUpgrade Configuration Files for Different Patching Scenarios

In the following configuration file examples, the following Oracle Databases are patched from Oracle Database 19c patched to Release Update (RU) 11 to 19c RU 13:

- A non-CDB database patched from 19.11.0 to 19.13.0, designated with the prefix `patch1`
- An Oracle Database 19c CDB patched from 19.11 to 19.13, designated with the prefix `patch2`
- An encrypted PDB patched by unplug-plug, designated with the prefix `patch3`
- A non-CDB, which is patched and converted to a CDB, designated with the prefix `patch4`
- A CDB with a PDB that is relocated during the patching operation, designated with the prefix `patch5`
- An Oracle Real Application Clusters (Oracle RAC) database, designated with the prefix `patch6`
- An Oracle RAC database in a distributed cluster configuration, designated with the prefix `patch7`

```
#
# Global log directory for patch logs
#
global.autoupg_log_dir=/databases/patchlogs
#
# Non-CDB patch to Non-CDB patch, source and target home
#
patch1.sid=db19
patch1.source_home=/databases/ee/product/1911/dbhome_1
patch1.target_home=/databases/ee/product/1913/dbhome_2
#
#
# CDB patch, Source and Target home
#
patch2.sid=cdb19
patch2.source_home=/databases/ee/product/1911/dbhome_1
patch2.target_home=/databases/ee/product/1913/dbhome_2
#
# Unplug-Plug with KeyStore
#
global.keystore=/databases/tde
patch3.sid=cdb19
patch3.source_home=/databases/ee/product/1911/dbhome_1
patch3.target_home=/databases/ee/product/1913/dbhome_2
patch3.target_cdb=cdb1913
patch3.target_pdb_name=sales
patch3.target_pdb_copy_option=file_name_convert=('/databases/ee/oradata/CDB19/
sales', '/databases/ee/oradata/CDB1913/sales')
#
# Non-CDB to CDB, Source and Target home
#
patch4.sid=db19
patch4.source_home=/databases/ee/product/1911/dbhome_1
patch4.target_home=/databases/ee/product/1913/dbhome_2
patch4.target_cdb=cdb1913
patch4.target_pdb_name=emp
```

```
patch4.target_pdb_copy_option=file_name_convert=('/databases/ee/oradata/
DB19', '/databases/ee/oradata/CDB1913/emp')
#
# Patch relocate DB
#
patch5.sid=cdb19
patch5.source_home=/databases/ee/product/1911/dbhome_1
patch5.target_home=/databases/ee/product/1913/dbhome_2
patch5.target_cdb=cdb1913
patch5.pdbs=cars
patch5.target_pdb_copy_option.cars=file_name_convert=('/databases/ee/oradata/
CDB19/cars', '/databases/ee/oradata/CDB1913/cars')
patch5.source_dblink.cars=db19_link
#
# Oracle RAC
#
patch6.sid=rac1
patch6.source_home=/databases/ee/product/1911/dbhome_1
patch6.target_home=/databases/ee/product/1913/dbhome_2
#
# Distributed Oracle RAC with proactive fixups
#
patch7.sid=rac2
patch7.source_home=/databases/ee/product/1911/dbhome_1
patch7.target_home=/databases/ee/product/1913/dbhome_2
patch7.tune_setting=distributed_upgrade=true
```

Example 4-10 A Summary Log File for AutoUpgrade Patching

In this patching summary report file, you can see how AutoUpgrade applies patches to the CDB and PDBs.

```
********************************************************************************
**
        Datapatch Apply Summary Report for CDB$ROOT

    Return code        = 0 SUCCESS
    Failure reason     = null
    Total time         = 161.721805095673
    Install patches    = 1
    Database Open      = SUCCESS
    Invocation Log     = /databases/cfgtoollogs/sqlpatch/
sqlpatch_17781_2022_05_18_10_57_58/sqlpatch_invocation.log
    Bootstrap Required = 1
    Bootstrap Status   = SUCCESS
    Bootstrap Log      = /databases/cfgtoollogs/sqlpatch/
sqlpatch_17781_2022_05_18_10_57_58/bootstrap1_CDB19X_CDBROOT.log
    Total patches      = 1

    Patch Key          = 33192793-24462514
    Mode               = apply
    Status             = SUCCESS
    Patch Log File     = /databases/cfgtoollogs/sqlpatch/
33192793/24462514/33192793_apply_CDB19X_CDBROOT_2022May18_10_58_06.log
    RU Log File        = /databases/cfgtoollogs/sqlpatch/
```

```
33192793/24462514/33192793_ru_apply_CDB19X_CDBROOT_2022May18_10_58_05.log
    RU Errors          = N/A

********************************************************************************
**
        Datapatch Apply Summary Report for PDBX

    Return code        = 0 SUCCESS
    Failure reason     = null
    Total time         = 123.969398021698
    Install patches    = 1
    Database Open      = SUCCESS
    Invocation Log     = /databases/cfgtoollogs/sqlpatch/
sqlpatch_18416_2022_05_18_11_01_40/sqlpatch_invocation.log
    Bootstrap Required = 1
    Bootstrap Status   = SUCCESS
    Bootstrap Log      = /databases/cfgtoollogs/sqlpatch/
sqlpatch_18416_2022_05_18_11_01_40/bootstrap1_CDB19X_PDBX.log
    Total patches      = 1

    Patch Key          = 33192793-24462514
    Mode               = apply
    Status             = SUCCESS
    Patch Log File     = /databases/cfgtoollogs/sqlpatch/
33192793/24462514/33192793_apply_CDB19X_PDBX_2022May18_11_01_55.log
    RU Log File        = /databases/cfgtoollogs/sqlpatch/
33192793/24462514/33192793_ru_apply_CDB19X_PDBX_2022May18_11_01_55.log
    RU Errors          = N/A

********************************************************************************
**
        Datapatch Apply Summary Report for PDB$SEED

    Return code        = 0 SUCCESS
    Failure reason     = null
    Total time         = 124.234117984772
    Install patches    = 1
    Database Open      = SUCCESS
    Invocation Log     = /databases/cfgtoollogs/sqlpatch/
sqlpatch_18406_2022_05_18_11_01_40/sqlpatch_invocation.log
    Bootstrap Required = 1
    Bootstrap Status   = SUCCESS
    Bootstrap Log      = /databases/cfgtoollogs/sqlpatch/
sqlpatch_18406_2022_05_18_11_01_40/bootstrap1_CDB19X_PDBSEED.log
    Total patches      = 1

    Patch Key          = 33192793-24462514
    Mode               = apply
    Status             = SUCCESS
    Patch Log File     = /databases/cfgtoollogs/sqlpatch/
33192793/24462514/33192793_apply_CDB19X_PDBSEED_2022May18_11_01_55.log
    RU Log File        = /databases/cfgtoollogs/sqlpatch/
33192793/24462514/33192793_ru_apply_CDB19X_PDBSEED_2022May18_11_01_55.log
    RU Errors          = N/A
```

AutoUpgrade Patching Workflow Examples

Use this example of an AutoUpgrade patchiing workflow to understand the AutoUpgrade patching procedure.

In this scenario, you use AutoUpgrade Patching to download the patch, deploy the patch, and start the patching operation.

Example 4-11 Configuration files for AutoUpgrade Patching

You can choose to have AutoUpgrade patching to download patches, or you can first download patches manually and then run AutoUpgrade patching.

In the following configuration file example, we set the downlload option for job `upg1` to `YES`:

```
global.global_log_dir=/logs/patching
global.keystore=/secure/keystore
upg1.sid=DB19X
upg1.source_home=/databases/19x/dbhome_1
upg1.target_home=/databases/19x/dbhome_2
upg1.folder=/storage/patches
upg1.download=YES
```

In the following configuration file example, we set the downlload option for job `upg1` to `NO`:

```
global.global_log_dir=/logs/patching
upg1.sid=DB19X
upg1.source_home=/databases/19x/dbhome_1
upg1.target_home=/databases/19x/dbhome_2
upg1.folder=/storage/patches
upg1.patch=RU,OPATCH
upg1.download=NO
```

Example 4-12 AutoUpgrade Patching Download Option

To enable AutoUpgrade patching to automatically download patches, you have to set the download parameter in the configuration file for the job running the patch. As AutoUpgrade runs, all identified patches will be automatically downloaded and verified. If a patch is already in the directory that you defined in the configuration file using the `folder` configuration parameter, then download of that patch will be skipped. If a proxy is needed to connect to My Oracle Support, then you can set the Linux environment variables `https_proxy`, `http_proxy` and `no_proxy` so that AutoUpgrade can connect to Oracle Support.

Complete this procedure:

1. Ensure that the configuration file uses the `download=YES` option, or leave this option unspecified; the default value will include it. The global keystore parameter will also be required.

   For example:

   ```
   global.global_log_dir=/logs/patching
   global.keystore=/secure/keystore
   . . .
   upg1.sid=DB19X
   upg1.source_home=/databases/19x/dbhome_1
   ```

```
upg1.target_home=/databases/19x/dbhome_2
upg1.folder=/patches
upg1.patch=RECOMMENDED
upg1.download=YES
```

2. Load the My Oracle Support credentials into the keystore by using the `-load_password` command line option:

   ```
   java -jar autoupgrade.jar -patch -config config.cfg -load_password
   ```

   When the password console loads, add the necessary My Oracle Support credentials by using the `add` option.

   For example:

   ```
   MOS> group MOS
   MOS> add -user user@company.com
       Enter your secret/Password:
       Re-enter your secret/Password:
   MOS> list
       MOS Credentials Loaded - Connection Successful
   ```

3. Run AutoUpgrade Patching in any of the supported modes: ANALYZE, FIXUPS, DEPLOY, DOWNLOAD, CREATE_HOME.

   For example, run the analyze mode:

   ```
   java -jar autoupgrade.jar -patch -config config.cfg -mode analyze
   ```

Example 4-13 AutoUpgrade Patching Deploy Option

After you have loaded My Oracle Support credentials into the keystore, you can patch the source database using the DEPLOY option.

In the following example, using the same configuration file (`config.cfg`) for the 19X database showin in the previous example, we deploy the patches with AutoUpgrade:

```
java -jar autoupgrade.jar -patch -config config.cfg -mode deploy
```

The console output for this example operation is as follows:

```
AutoUpgrade Patching 24.1.240816 launched with default internal options
Processing config file ...
Loading AutoUpgrade Patching keystore

----------------------------
Downloading files to /patches
----------------------------
DATABASE RELEASE UPDATE 19.24.0.0.0
    File: p36582781_190000_Linux-x86-64.zip - VALIDATED

DATAPUMP BUNDLE PATCH 19.24.0.0.0
    File: p36682332_1924000DBRU_Generic.zip - VALIDATED

OJVM RELEASE UPDATE 19.24.0.0.0
    File: p36414915_190000_Linux-x86-64.zip - VALIDATED
```

```
   OPatch 12.2.0.1.43 for DB 19.0.0.0.0 (Jul 2024)
       File: p6880880_190000_Linux-x86-64.zip - VALIDATED
   ----------------------------

   +----------------------------------------+
   | Starting AutoUpgrade Patching execution |
   +----------------------------------------+
   1 Non-CDB(s) will be processed
   Type 'help' to list console commands
   patch> Job 100 completed
   ------------------ Final Summary --------------------
   Number of databases            [ 1 ]

   Jobs finished                  [1]
   Jobs failed                    [0]
   Jobs restored                  [0]
   Jobs pending                   [0]

   ---- Drop GRP at your convenience once you consider it is no longer needed
----
   Drop GRP from DB19X: drop restore point AU_PATCHING_9212_DB19X1919000


   Please check the summary report at:
   /logs/patching/cfgtoollogs/patch/auto/status/status.html
   /logs/patching/cfgtoollogs/patch/auto/status/status.log
```

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.

- Create Configuration File for AutoUpgrade
  To use AutoUpgrade to complete the upgrade, you first create a configuration file with AutoUpgrade from the new release Oracle home.
- 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.
- AutoUpgrade Configuration File For Upgrading PDBs Already in the Target CDB
  See how you can specify an upgrade of earlier release PDBs that are on a later release 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.
- Unplug-Plug Relocate Upgrades With AutoUpgrade
  See how you can use the Unplug-Plug Relocate feature (also known as Hot-Cloning upgrade) to create PDBs that can be refreshed for a given period before upgrading them.
- Ignore Fixups and Checks Using the AutoUpgrade Configuration File
  To skip an entire check and fixup step for a database, you can direct AutoUpgrade to read the fixup `runfix` flag from the fixups checklist file, and set the flag for that fixup from `YES` to `SKIP`.
- Run Custom Scripts Using AutoUpgrade
  Learn how to use AutoUpgrade to run your own scripts as part of the deploy process.

Create Configuration File for AutoUpgrade

To use AutoUpgrade to complete the upgrade, you first create a configuration file with AutoUpgrade from the new release Oracle home.

In the following example, the AutoUpgrade utility is run using the parameter `sample_config_file`. This parameter generates a configuration file in the home of the user running AutoUpgrade that you can edit to provide environment paths and settings and upgrade preferences for the upgrade. To generate the configuration file (`config`), you run AutoUpgrade from the new release Oracle Database home using the `sample_config_file` parameter, and specify an output file name.

> **Note:**
>
> AutoUpgrade is regularly updated. For additional examples, and for information about the most recent AutoUpgrade releases, including new command-line parameters and options, and new or enhanced configuration file parameters, refer to the Oracle Database Upgrade Guide for the release to which you want to upgrade. Also refer to the My Oracle Support note "AutoUpgrade Tool (Doc ID 2485457.1)," which will contain information about the most recent AutoUpgrade updates.

In this example, user `oracle` navigates to the location of an earlier release Oracle home, which in this example is Oracle Database 19c:

```
cd /u01/app/oracle/product/19.0.0/
```

Next, the Oracle user starts AutoUpgrade from the Oracle Database 23ai Oracle home, and creates a configuration file in its user home directory, `/home/oracle`:

```
java -jar /u01/app/oracle/product/23/rdbms/admin/autoupgrade.jar -
create_sample_file config
Created sample configuration file /home/oracle/sample_config.cfg
```

After you create the configuration file, open it up in your preferred text editor, and modify parameter settings as needed for your environment.

```
cd /
vi sample_config.cfg
```

Related Topics

- Oracle Database Documentation
- AutoUpgrade Tool (Doc ID 2485457.1)

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 `/u01/app/oracle/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=/u01/app/oracle/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/19.0.0/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/19.0.0/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
```

On the Oracle Database Upgrades and Migration YouTube Channel, you can see a similar upgrade scenario, *Oracle Database AutoUpgrade 19c - Upgrading 2 databases in parallel* (11:57), demonstrated by Mike Dietrich.

Related Topics

- Oracle Database AutoUpgrade 19c - Upgrading 2 databases in parallel

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.

AutoUpgrade Configuration File For Upgrading PDBs Already in the Target CDB

See how you can specify an upgrade of earlier release PDBs that are on a later release CDB.

In this scenario, you upgrade specific PDBs that were excluded in a previous upgrade of the CDB and PDBs, or PDBs that were added to the CDB through manual Non-CDB to PDB or Unplug-Plug operations. The PDBs are in an earlier release version.

The configuration file has a Database 1 section that identifies the CDB database as `CDB19X`, which is the CDB that has the existing PDBs `PDB1` and `PDB2`. In this scenario CDB19X has 10 PDBs, PDB1 through PDB10, but PDB1 and PDB2 were excluded from an earlier upgrade procedure. CDB19X and PDBs 3 through 10 are all upgraded to the later release, but PDB1 and PDB2 remain on an earlier release. We will now run AutoUpgrade to upgrade the remaining two PDBs on the earlier release, specifying that the target upgrade PDBs `PDB1` and `PDB2` are on `CDB19X` (`upg1.sid=CDB19X`), and the upgrade should start immediately with `upg.start_time=now`.

> **Note:**
>
> Before you begin, the PDBs PDB1 and PDB2 must already be in Upgrade mode.

Update the configuration file (`config.cfg`):

```
#
# Global logging directory pertains to all jobs
#
global.autoupg_log_dir=/home/oracle/autoupg
#
# Database 1
#
upg1.sid=CDB19X
upg1.start_time=now
upg1.target_home=/u01/app/oracle/product/19.0.0/dbhome_1
upg1.pdbs= PDB1,PDB2
```

Start the upgrade with AutoUpgrade:

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

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 4-14 Running a script using `noconsole` mode

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.

Unplug-Plug Relocate Upgrades With AutoUpgrade

See how you can use the Unplug-Plug Relocate feature (also known as Hot-Cloning upgrade) to create PDBs that can be refreshed for a given period before upgrading them.

You can use the Unplug-Plug relocate upgrade as a method to create clones from other PDBs, or from non-CDB databases. If the source database is a non-CDB, then AutoUpgrade upgrades and converts the non-CDB as part of the upgrade process. This feature is compatible with Oracle Database 12c release 1 (12.1.0.2) and later releases as the source release. Starting with AutoUgrade version 22.5, AutoUpgrade supports unplug-relocate operations using a source of at least Oracle Database 12c Release 1 (12.1.0.2). When a 12.1.0.2 source is used, the deploy operation requires the source to be in read only mode, and the target CDB to not be in local undo mode. The `CLONE_ONLY` option is also required when specifying the `source_dblink parameter`.Oracle Database Release 11g (11.2) is not compatible as a source version for the AutoUpgrade Hot-Cloning feature.

To use the Unplug-Plug relocate upgrade method, the following configuration must be complete:

- On the database that you plan to clone, a user must exist that is granted the `CREATE SESSION`,`CREATE PLUGGABLE DATABASE`, `SELECT_CATALOG_ROLE`, and `GRANT READ ON sys.enc$` privileges.
- On the CDB where you plan to relocate the PDB, you must create a database link (`@dblink`) on the CDB that points to the database that you plan to clone.
- The target CDB must have configured all of the components that the source database has configured, or the upgrade process will fail.

Make a note of the database link, because you need it to fill out the AutoUpgrade configuration file.

In the following example, we will go through each step in the process, for both a non-CDB and a PDB. The scenario below is composed of two hosts:

- `db-node1`: The source database host location, where step 1 is run.
- `db-node2`: The target database host location, where step 2 is run.

> **Note:**
>
> This example uses two separate hosts, but this procedure can also be run on a single host.

1. Grant a user on the database that you want to clone the `CREATE SESSION`, `CREATE PLUGGABLE DATABASE`, and `SELECT_CATALOG_ROLE` privileges. Also, on SYS, give the user `GRANT READ ON sys.enc$`. In this example, the user `clone` is created with these privileges,

   - Non-CDB:

     ```
     alter system set local_listener='(ADDRESS=(PROTOCOL=TCP)(HOST=db-node1)
     (PORT=1521))';
     ALTER SYSTEM REGISTER;
     CREATE USER clone IDENTIFIED BY some-password;
     GRANT CREATE SESSION, CREATE PLUGGABLE DATABASE, SELECT_CATALOG_ROLE TO
     clone;
     GRANT READ ON sys.enc$ TO user;
     ```

   - PDB:

     ```
     alter system set local_listener='(ADDRESS=(PROTOCOL=TCP)(HOST=db-node1)
     (PORT=1521))';
     ALTER SYSTEM REGISTER;
     CREATE USER c##clone IDENTIFIED BY some-password CONTAINER=ALL;
     GRANT CREATE SESSION, CREATE PLUGGABLE DATABASE, SELECT_CATALOG_ROLE TO
     c##clone CONTAINER=ALL;
     GRANT READ ON sys.enc$ TO c##clone CONTAINER=ALL;
     ```

     On a PDB, you can also limit the scope of creating the user to a single PDB on the CDB by specifying a particular PDB name. For example:

     ```
     CREATE USER c##clone IDENTIFIED BY some-password CONTAINER=PDBX;
     ```

2. Configure the CDB to which you want to clone the PDB by creating a database link pointing to the source database on the source system that you plan to clone.

   In the following example, the database link points to the source database `db18x`:

   - Non-CDB

     In the following example, the database link points to the source database `db18x`:

     ```
     CREATE DATABASE LINK db18x_link
     CONNECT TO clone IDENTIFIED BY some-password
     USING'(DESCRIPTION =(ADDRESS = (PROTOCOL = TCP)(HOST = db-node1)(PORT =
     1521))(CONNECT_DATA = (SERVICE_NAME = db18x)))';
     ```

   - PDB

     In the following example, the database link points to the source PDB `pdbx` on the CDB `cdb18x`:

     ```
     CREATE DATABASE LINK pdbxcdb18x_link
     CONNECT TO c##clone IDENTIFIED BY some-password
     ```

```
USING'(DESCRIPTION =(ADDRESS = (PROTOCOL = TCP)(HOST = db-node1) (PORT =
1521))(CONNECT_DATA = (SERVICE_NAME = pdbx)))';
```

3. Update the AutoUpgrade configuration file with the information needed to identify the databases that you want to clone.

   Each database that you plan to clone must have the AutoUpgrade local parameter `target_pdb_copy_option` set to convert file names.

   > **Note:**
   >
   > 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`. However, you must have a file name and path specified either in the configuration file, or on the target CDB, or the upgrade will fail.

   By using the prefix for local parameters for each database, you can combine non-CDB and PDB upgrades in the same configuration file to relocate both non-CDBs and PDBs to the same target CDB. The following is an example of the relevant portion of the configuration file where both the non-CDB database `db18x` is configured for upgrade, and the PDB `pdbx` is configured for upgrade. In this case, file names for `db18x` are set to change the prepended name from `DB18x` to `db18x`, and the file names for the :

   ```
   upg1.sid=db18x
   upg1.source_home=/source/18x
   upg1.target_home=/target/19x
   upg1.target_cdb=cdb19x
   upg2.sid=cdb18x
   upg2.pdbs=pdbx
   upg2.target_cdb=cdb19x
   upg2.source_home=/source/18x
   upg2.target_home=/target/19x
   ```

   Each database has a `target_pdb_copy_option` set. For example, here the non-CDB filenames are converted from the prepended string `DB18X` to `db18x`:

   ```
   upg1.target_pdb_copy_option.db18x=file_name_convert=('DB18X', 'db18x')
   ```

   Each database must have the local parameter `source_dblink` set to the database link that you created on the target CDB:

   ```
   upg1.source_dblink.db18x=db18x_link
   ```

   To improve consistency, you can also use the local parameter `start_time` in combination with `source_dblink` with the optional refresh rate (in seconds) to set a refresh rate for the data, and schedule the job start time. The following parameters combined start AutoUpgrade, but sets a delay for the deployment of the upgrade of one hour and 40

minutes from the time that AutoUpgrade was started, and sets a refresh rate from the source database files to the target database files every 20 seconds:

```
upg1.source_dblink.db18x=db18x_link 20
upg1.start_time=+1h40m
```

If you do not set a refresh rate for `source_dblink`, then the database files of the cloned database are cloned only once, without a refresh. If `start_time` is not set (so there is no delay in the processing of the upgrade), then the `source_dblink` refresh rate value is ignored.

4. Start the Unplug-Plug relocate deployment, specifying the configuration file you created for this task:

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

Example 4-15 Configuration File for Unplug-Plug Relocate of Non-CDB and PDB with No Refresh Rate for Data Files

```
global.autoupg_log_dir=/home/oracle/xupg
#database1
upg1.sid=cdb18x
upg1.target_cdb=cdb19x
upg1.source_home=/databases/ee/product/18x/dbhome_1
upg1.target_home=/databases/ee/product/19x/dbhome_1
upg1.pdbs=pdbx
upg1.target_pdb_name.pdbx=pdbxr
upg1.target_pdb_copy_option.pdbx=file_name_convert=('pdbx', 'pdbxr')
upg1.source_dblink.pdbx=pdbxcdb18x_link
#database2
upg2.sid=db18x
upg2.target_cdb=cdb19x
upg2.source_home=/databases/ee/product/18x/dbhome_1
upg2.target_home=/databases/ee/product/19x/dbhome_1
upg2.target_pdb_copy_option.db18x=file_name_convert=('DB18X', 'db18x')
upg2.source_dblink.db18x=db18x_link
upg2.target_pdb_name.db18x=db18x
```

Example 4-16 Configuration File for Unplug-Plug Relocate of Non-CDB and PDB with Deployment Delay and Refresh Rate for Data Files

```
global.autoupg_log_dir=/home/oracle/xupg
#database1
upg1.sid=cdb18x
upg1.target_cdb=cdb19x
upg1.source_home=/databases/ee/product/18x/dbhome_1
upg1.target_home=/databases/ee/product/19x/dbhome_1
upg1.pdbs=pdbx
upg1.target_pdb_name.pdbx=pdbxr
upg1.target_pdb_copy_option.pdbx=file_name_convert=('pdbx', 'pdbxr')
upg1.source_dblink.pdbx=pdbxcdb18x_link 600
upg1.start_time=+3h
#database2
upg2.sid=db18x
upg2.target_cdb=cdb19x
```

```
upg2.source_home=/databases/ee/product/18x/dbhome_1
upg2.target_home=/databases/ee/product/19x/dbhome_1
upg2.target_pdb_copy_option.db18x=file_name_convert=('DB18X', 'db18x')
upg2.source_dblink.db18x=db18x_link 900
upg2.target_pdb_name.db18x=db18x
upg2.start_time=+3h30m
```

Example 4-17 Simple Configuration File for Unplug-Plug Relocate of a PDB

```
global.autoupg_log_dir=/home/oracle/xupg #database1
upg1.sid=cdb18x
upg1.target_cdb=cdb19x
upg1.source_home=/databases/ee/product/18x/dbhome_1
upg1.target_home=/databases/ee/product/19x/dbhome_1
upg1.pdbs=pdbx
upg1.target_pdb_copy_option.pdbx=file_name_convert=('pdbx', 'pdbxr')
upg1.source_dblink.pdbx=pdbxcdb18x_link
```

For more examples of unplug-plug upgrades, see Mike Dietrich's blog, "Upgrade Your Database Now!" and Daniel Overby Hansen's blog, "Databases are Fun". New use case examples are regularly presented on these sites. Two examples are appended below.

Related Topics

- Unplug/Plug Upgrade with AutoUpgrade
- Upgrading an Encrypted PDB

Ignore Fixups and Checks Using the AutoUpgrade Configuration File

To skip an entire check and fixup step for a database, you can direct AutoUpgrade to read the fixup `runfix` flag from the fixups checklist file, and set the flag for that fixup from `YES` to `SKIP`.

To override the default list of fixups that AutoUpgrade performs automatically for upgrades, you can load an existing checklist of fixup steps in to AutoUpgrade by using the local configuration file parameter `checklist`. In the checklist that you specify with the local parameter, you can set a precheck fixup as follows:

- `YES` (the default): Run checks, and run fixups
- `NO`: Run checks, but do not run fixups.
- `_SKIP`: Do not run checks, and do not run fixups.

In the configuration file, the local parameter checklist is used to direct AutoUpgrade to an existing checklist file.

```
global.autoupg_log_dir=/home/oracle/autoupg

upg1.sid=db12204
upg1.source_home=/databases/ee/product/12.2.0/dbhome_1
upg1.target_home=/databases/ee/product/21.1.0/dbhome_1
upg1.checklist=/home/oracle/autoupg/db12204/100/prechecks/
db11204_checklist.cfg
```

In the Fixups checklist file, the `runfix` flag for the `DICTIONARY_STATS` fixup is is set to skip that step.

```
[SID] [db21.1.0]
========================================
[container] [db21.1.0]
========================================
[checkname] EXISTENCE_OF_DATAPUMP_AQ_TABLES
[stage] PRECHECKS
[fixup_available] YES
[runfix] YES
[severity] WARNING
--------------------------------------------------
[checkname] DICTIONARY_STATS
[stage] PRECHECKS
[fixup_available] YES
[runfix] _SKIP
[severity] RECOMMEND
--------------------------------------------------
```

Run Custom Scripts Using AutoUpgrade

Learn how to use AutoUpgrade to run your own scripts as part of the deploy process.

You can configure four parameters in the configuration file that enable you to run custom scripts:

- `before_action`

  Run a custom script on the source Oracle home before the database is upgraded, during the Preupgrade stage when the database is up.

- `after_action`

  Run a custom script on the target Oracle home after the database is upgraded, during the Postupgrade stage or PDB upgrade stage, depending on the options set in the configuration file.

- `revert_before_action`

  Run a custom script on the target Oracle home before the database is restored, typically when the database is still up. Due to the nature of the upgrade process, be aware that the database can be down when the script is started. You can use this script to undo actions performed by a `before_action` custom script.

- `revert_after_action`

  Run a custom script on the source Oracle home after the database restored, and the database is up. You can use this script to undo actions performed by an `after_action` custom script.

See parameter descriptions under "AutoUpgrade Utility Configuration Files" for more detailed information about these parameters.

Example 4-18 Using `before_action` and `after_action` in the Configuration File

In the following example of configuration file `config.cfg`, the local parameter `before_action` is used in the configuration file to run the script `script1.sh` in the path `/path/to/a/` before the

restoration, and the local parameter `after_action` is used to run the script `script2.sh` in the path `/path/to/a/` after the upgrade.

```
config.cfg

global.autoupg_log_dir=/home/oracle/autoupg
upg1.sid=db12204
upg1.source_home=/databases/ee/product/12.2.0/dbhome_1
upg1.target_home=/databases/ee/product/19.0.0/dbhome_1
upg1.before_action=/path/to/a/script1.sh
upg1.after_action=/path/to/a/script2.sh
```

The configuration file is called by AutoUpgrade from the command line:

```
java -jar autoupgrade.jar -config config.cfg -restore -jobs 100,101,...
```

Because these scripts are run during the preupgrade and postupgrade stages, restoration to the GRP occurs before or after the scripts are run. The scripts do not affect the actual GRP process itself, which makes them safer for reruns if the script actions can affect the database.

Example 4-19 Using `revert_before_action` and `revert_after_action` in the Configuration File

In the following example of configuration file `config.cfg`, the local parameter `revert_before_action` is used in the configuration file to run the script `script3.sh` in the path `/path/to/a/` before the restoration, and the local parameter `revert_after_action` is used to run the script `script4.sh` in the path `/path/to/a/` after the upgrade.

```
global.autoupg_log_dir=/home/oracle/autoupg
upg1.sid=db12204
upg1.source_home=/databases/ee/product/12.2.0/dbhome_1
upg1.target_home=/databases/ee/product/19.0.0/dbhome_1
upg1.revert_before_action=/path/to/a/script3.sh
upg1.revert_after_action=/path/to/a/script4.sh
```

Again, the configuration file is used to direct AutoUpgrade as it runs the restore jobs:

```
java -jar autoupgrade.jar -config config.cfg -restore -jobs 100,101,...
```

In this case, the scripts are run on the target Oracle Database binaries when the database is up. The `script3.sh` script called by `revert_before_action` is run when the database is up, before starting the restoration to the GRP, and the `script4.sh` script called by `revert_after_action` is run on the database after restoration to the earlier release is complete, and the database is up in the original Oracle home.

Related Topics

- AutoUpgrade Utility Configuration Files

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 4-3 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.
`lsjNoConsoleTimer`	30	Runtime interval in seconds of the `lsj` command for `noconsole`. The interval should be between 7 and 1200 seconds.
`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.
`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.

Table 4-3 (Cont.) Internal Settings Configuration File Parameters for AutoUpgrade

Parameter	Default	Description
dbPostUpgradeAbortTimer	60	Number of minutes to wait before the postupgrade 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.autoupg_log_dir=/database/jobmgr
```

AutoUpgrade configuration and status file paths are relative to the directory path that you establish with `global.autoupg_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 4-20 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 4-21 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 4-22 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 4-23 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.

Known Restrictions for AutoUpgrade

If you encounter issues with your upgrade, review the known restrictions to find solutions.

- AutoUpgrade and Disk Space Issues
  If you run out of disk space while running AutoUpgrade, then review and apply the solutions outlined here.
- AutoUpgrade and OracleServiceAutoUpgradeSid
  The ability to use the Microsoft Windows service `OracleServiceAutoUpgradeSid` for patching and upgrading is deprected.

AutoUpgrade and Disk Space Issues

If you run out of disk space while running AutoUpgrade, then review and apply the solutions outlined here.

To assist you to avoid disk space issues. Oracle has added a check to calculate the required disk space. To see log sizing specifications, review the AutoUpgrade analyze reports. If the database runs out of disk space during the upgrade, then free up the required disk space for the database, and resume the job, if necessary.

Example 4-24 Disk Space Errors While Upgrade is Running

1. Stop the current running upgrade

   ```
   abort -job 100
   ```

2. Free up disk space so that the upgrade can resume.

3. Resume the stopped upgrade

```
resume -job 100
```

Example 4-25 Disk Space Errors After Upgrade has Completed

If AutoUpgrade completes with disk space errors then proceed as follows:

1. Free up disk space so that the upgrade can resume.
2. Restart AutoUpgrade to complete any remaining issues.

   For example, where the configuration file name is *yourconfigfile.cfg*:

```
java -jar autoupgrade.jar -mode deploy -config yourconfigfile.cfg
```

AutoUpgrade and OracleServiceAutoUpgradeSid

The ability to use the Microsoft Windows service `OracleServiceAutoUpgradeSid` for patching and upgrading is deprected.

In previous releases, while it was not documented, it was possible to create the Microsoft Windows service `OracleServiceAutoUpgradeSid` for patching and upgrading. This capability will be removed. In future releases, AutoUpgrade will only support the creation of Microsoft Windows services for Oracle that require passwords to use the AutoUpgrade `load_win_credential` parameter. When this parameter is used, AutoUpgrade will then be able to create Oracle Services with the correct password in a secure manner, and provide a seamless operation for database administrators.

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 `clear_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 -clear_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 Keystore With Credentials Not Set With AUTOLOGIN

Problem Description: You run the tool, and you have keystore 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 keystore credentials unless they are configured for automatic logins into the database

Related Topics

- Performing Operations That Require a Keystore Password

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 4-26 Starting Up the AutoUpgrade Utility in Analyze Mode

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

Example 4-27 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 4-28 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).
`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
```

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

5

Upgrading Oracle Database Using Parallel Upgrade Utility or Replay Upgrade

You can upgrade manually by using the Parallel Upgrade Utility command-line option, or you can use the Replay Upgrade process.

Starting with Oracle Database 21c, 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.
- About Transporting and Upgrading a Database (Full Transportable Export/Import)
  You can use file-based or nonfile-based modes for transporting data.
- Upgrading Oracle Database Releases Using Replay Upgrade
  To upgrade from an earlier release, you can use the Oracle Multitenant Replay Upgrade (Replay Upgrade) procedure to adopt a non-CDB to a PDB, or upgrade a PDB.
- Manual Non-CDB Oracle Database Release Upgrades to Multitenant Architecture
  To manage your non-CDB Oracle Database upgrade manually by using scripts, learn about upgrade scenarios and procedures.
- 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.
- Rerunning Upgrades for Oracle Database
  Use these options to rerun upgrades.

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.

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 AutoUpgrade using the `preupgrade` clause 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. Run AutoUpgrade with the `preupgrade` parameter on the source database, and correct any issues that AutoUpgrade does not fix.
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.

   ```
   SQL> startup upgrade;
   SQL> alter pluggable database all open 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 5-1 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, you want to include PDB1 and PDB2, but exclude other PDBs:
	Linux and Unix (use single quotes):
	`-c 'PDB1 PDB2'`
	Windows (use double quotes):
	`-c "PDB1 PDB2"`
	As a result of this specification, PDB1 and PDB2 are upgraded, but PDB3 and PDB4 are not upgraded.

-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:
	Linux and Unix (use single quotes):
	`-C 'PDB1 PDB2'`
	Windows (use double quotes):
	`-C "PDB1 PDB2"`
	As a result of this specification, PDB1 and PDB2 are not upgraded, but PDB3 and PDB4 are upgraded.
	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.

Table 5-1 (Cont.) Parallel Upgrade Utility (catctl.pl) Parameters

Parameter	Description
-F	Forces a cleanup of previous upgrade errors. 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.
-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$ROOT in UPGRADE mode while the PDBs are upgraded. During upgrades, using this parameter setting places the CDB and all its PDBs in upgrade mode, which can reduce total upgrade time. However, you cannot bring up any of the PDBs until the CDB and all its PDBs are upgraded. By default, if you do not use the -M parameter setting, then CDB$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. The number of PDBs upgraded concurrently is controlled by the value of the -n parameter. Multiple PDB upgrades are processed together. The default value 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: 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. The maximum value is 8. The minimum value is 1. The default value is 2.
-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.
-t	Uses classic upgrade for the upgrade, instead of the default Replay Upgrade process.
-T	Takes offline user schema-based table spaces.

Table 5-1 (Cont.) Parallel Upgrade Utility (catctl.pl) Parameters

Parameter	Description
`-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`, and Replay Upgrade to upgrade the PDBs. Replay Upgrade is the default option for upgrading PDBs starting with Oracle Database 21c. 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 5-1 Running Parallel Upgrade Utility with Parameters

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 5-2 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)

  > **Note:**
  >
  > If you are using Replay Upgrade, then the number is derived from `CPU_COUNT` divided by 4, so the number is 3.

- 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 21c, upgrades are supported only with using the multitenant architecture. Multitenant architecture enables Oracle Database deployments using multitenant container databases (CDB) that contain pluggable databases (PDBs). For information about the number of PDBs you are permitted in a CDB for each deployment option, refer to *Oracle Database Licensing Information User Manual*.

- 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
  uou 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.
- Oracle Label Security Integration in a Multitenant Environment
  You can use Oracle Label Security in a multitenant environment.
- Upgrading Multitenant Architecture In Parallel
  Use this technique to upgrade multitenant architecture Oracle Database releases supported for direct upgrade 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), you can unplug the PDBs from the earlier release CDB, and plug the PDBs into the later release CDB.

About Oracle Multitenant Oracle Database Upgrades

You can upgrade Oracle Databases installed on multitenant architecture either in parallel, or in sequence.

You can upgrade multitenant architecture systems by using AutoUpgrade, or Oracle Restart upgrade, or by using Parallel Upgrade Utility to carry out manual upgrades. Starting with Oracle Database 21c, Replay Upgrade is the default option for upgrading PDBs.

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 run AutoUpgrade preupgrade scripts to prepare one or more PDBs to upgrade, and shut them down. You then unplug PDBs, plug them into the new release multitenant 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 AutoUpgrade with the `preupgrade` parameter

1. If you have not done so, run AutoUpgrade using the preupgrade clause. Review the 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, use SQL*Plus to 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.

   ```
   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 Microsoft 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\21.0.0\DB`, then enter the following command:

      ```
      C:\> ORADIM -NEW -SID ORCL -SYSPWD TWxy5791 -MAXUSERS 10
      -STARTMODE AUTO -PFILE C:\ORACLE\PRODUCT\21.0.0\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 target Oracle home.

c. The following environment variables point to the target Oracle home 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/21.0.0/dbhome_1/rdbms/admin
   $ sqlplus
   ```

   On Microsoft 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 or Unix:

    ```
    cd $ORACLE_HOME/bin
    ./dbupgrade
    ```

    Microsoft Windows:

    ```
    cd %ORACLE_HOME%\bin
    dbupgrade
    ```

    > **Note:**
    >
    > - 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 Microsoft 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 Microsoft 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. (Optional) Run `catcon.pl`. This command starts `utlrp.sql` and recompiles any remaining stored PL/SQL and Java code. You can recompile invalid objects manually, or let AutoUpgrade's automated postfixups recompile them for you in the next step.

    If you are recompiling code in one PDB at a time, then run the following command:

```
$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.

    If you are recompiling code in multiple PDBs at a time, then see the informational message in the preupgrade output for the syntax that Oracle recommends that you use. The recommended recompilation syntax can vary by platform.

18. Run the AutoUpgrade utility (`autoupgrade.jar`) with the option `-preupgrade`using the mode `postfixups`.

    For example:

```
java -jar autoupgrade.jar -preupgrade "dir=/
tmp,inclusion_list=PDB1,target_home=/databases/product/19c/dbhome_1" -mode
postfixups
```

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 utlu21s -d '''.''' utlusts.sql
```

Because the command specifies `-b utlu21s`, the upgrade results are stored in log file `utlu21s0.log`.

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:

    ```
    EXIT
    ```

22. (Conditional) If you are upgrading an Oracle Real Application Clusters database, then use the following command syntax in SQL*Plus to alter the system:

    ```
    ALTER SYSTEM SET CLUSTER_DATABASE=TRUE SCOPE=SPFILE;
    ```

23. Use `srvctl` to upgrade the database configuration in Oracle Clusterware:

    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.

    ```
    srvctl upgrade database -db db-unique-name -oraclehome oraclehome
    srvctl enable database -db db-unique-name
    ```

24. Use SQL*Plus to shut down the database:

    ```
    SHUTDOWN IMMEDIATE
    ```

25. Restart the database using `srvctl`:

    ```
    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 Database environment, relocate any data files in that environment to the new Oracle Database environment.

Related Topics

- *Oracle Database Administrator's Guide*

> **See Also:**
>
> *Oracle Database Administrator's Guide* for information about relocating data files

About Upgrading PDBs Using the Parallel Upgrade Utility with Priority Lists

uou 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 `MyUpgrade`, enter the following command:

```
C:>\u01\app\21.1.0\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 5-3 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 5-4 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 5-5 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 5-6 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 5-7 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 5-8 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.

Oracle Label Security Integration in a Multitenant Environment

You can use Oracle Label Security in a multitenant environment.

> **Note:**
>
> A multitenant container database is the only supported architecture in Oracle Database 21c and later releases. While the documentation is being revised, legacy terminology may persist. In most cases, "database" and "non-CDB" refer to a CDB or PDB, depending on context. In some contexts, such as upgrades, "non-CDB" refers to a non-CDB from a previous release.

In a multitenant environment, pluggable databases (PDBs) can be plugged in and out of a multitenant container database (CDB) or an application container.

- `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*

Upgrading Multitenant Architecture In Parallel

Use this technique to upgrade multitenant architecture Oracle Database releases supported for direct upgrade 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. For upgrades using Replay Upgrade (the default), 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), divided by 2. For classic upgrade, the maximum PDB upgrades that run concurrently are the number of PDBs that are upgraded in parallel, 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] [-t]

- `-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.

  For classic upgrade, the maximum PDB upgrades running concurrently is the value of `-n` divided by the value of `-N` for classic upgrade, and for the Replay Upgrade default, the value of -n divided by -N, divided by 2.
- `-t` Specifies that you want to use classic upgrade, using AutoUpgrade, instead of using the Replay Upgrade default.

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 AutoUpgrade using the `preupgrade` clause, to determine if there are any issues that you must correct before starting an upgrade. Fix any issue that is reported.

   For example:

   ```
   java -jar autoupgrade.jar -preupgrade "dir=/tmp,oh=/u01/app/product/12.2.0/
   dbhome_1,sid=db122,target_version=21" -mode analyze
   ```

3. Run the AutoUpgrade 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 5-9 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:

For Replay Upgrade, 6 PDBs are upgraded in parallel ([`CPU_COUNT` divided by 2], divided by 2, or 12 divided by 2). There are two parallel SQL processes allocated for each PDB.

For classic upgrade, 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 5-10 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:

For Replay Upgrade, 8 PDBs are upgraded in parallel ([64 divided by 4] divided by 2). There are 4 parallel SQL processes for each PDB.

For classic upgrade, 16 PDBs are upgraded in parallel (64 divided by 4). There are 4 parallel SQL processes for each PDB.

Example 5-11 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:

For Replay Upgrade, 5 PDBs are upgraded in parallel ([20 divided by 2], divided by 2). There are 2 parallel SQL processes for each PDB.

For classic upgrade, 10 PDBs are upgraded in parallel (20 divided by 2) There are 2 parallel SQL processes for each PDB.

Example 5-12 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:

For Replay Upgrade, 1 PDB is upgraded ([10 divided by 4], divided by 2). There are 4 parallel SQL processes for the PDB.

For classic upgrade, 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 to complete your upgrade, or to use Replay Upgrade, 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. To check readiness for upgrade, run AutoUpgrade using the `preupgrade` parameter, and use the `dir` option to specify an output log directory.

   ```
   java -jar autoupgrade.jar -preupgrade "dir=/tmp,oh=/u01/app/product/12.2.0/
   dbhome_1,sid=db122,target_version=21" -mode analyze
   ```

4. Check the `upgrade.xml` file in the log directory.

On multitenant architecture Oracle Databases, running AutoUpgrade using the `preupgrade` parameter with `fixups` mode runs fixups on every container that was open when you ran AutoUpgrade. The scripts resolve some issues that AutoUpgrade identifies.

Complete any other preupgrade tasks identified in the `upgrade.xml` file.

5. (Conditional) For Oracle RAC databases, use SQL*Plus to set the cluster database initialization parameter to `FALSE`:

   For example;

   ```
   ALTER SYSTEM SET cluster_database=FALSE SCOPE=spfile;
   ```

6. Shut down the database in the old Oracle home using `srvctl`.

   For example, where *`db_unique_name`* is your database name:

   ```
   cd $ORACLE_HOME
   pwd
   /u01/app/oracle/19.0.0/dbhome_1

   srvctl stop database -d db_unique_name
   srvctl disable database -d db_unique_name
   ```

7. Copy the `PFILE` or `SPFILE` from the old Oracle home to the new Oracle home
8. Connect with SQL*Plus:

   ```
   sqlplus / as sysdba
   ```

9. Bring the `CDB$ROOT` instance into upgrade mode:

   ```
   STARTUP UPGRADE
   ```

10. Bring all PDBs into upgrade mode:

    ```
    ALTER PLUGGABLE DATABASE ALL OPEN UPGRADE;
    ```

11. 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`.

12. Exit from SQL*Plus, and change directory to the new Oracle home `$ORACLE_HOME/rdbms/admin`:

    ```
    EXIT
    cd $ORACLE_HOME/rdbms/admin
    ```

13. 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
    ```

Starting with Oracle Database 21c, by default the `dbupgrade` script calls a Replay Upgrade, which sets parallelism to the number of CPUs divided by four. The number of PDBs upgraded in parallel is always half of the value previously used with legacy upgrade. On a server with 64 CPUs, 64 divided by 4 equals 16 PDBs upgraded in parallel.

> **Note:**
>
> If you prefer to use classic upgrade to perform the upgrade, then start `dbupgrade` using the `-t` option. For example:
>
> ```
> dbupgrade -t -d $ORACLE_HOME/rdbms/admin
> ```
>
> If you prefer to use AutoUpgrade, then refer to the AutoUpgrade script instructions.

14. Confirm that the upgrade was successful by reviewing the `upg_summary.log` If necessary, review other logs.
15. Open all PDBs using SQL*Plus, so that you can recompile the databases:

    ```
    ALTER PLUGGABLE DATABASE ALL OPEN;
    ```

16. Exit from SQL*Plus, and change directory to the new Oracle home path `$ORACLE_HOME/rdbms/admin`:

    ```
    EXIT
    cd $ORACLE_HOME/rdbms/admin
    ```

17. Run AutoUpgrade with the `preupgrade` parameter, run in `postfixups` mode. AutoUpgrade runs all database checks, and on the basis of those results, runs fixups automatically.

    For example:

    ```
    java -jar autoupgrade.jar -preupgrade "dir=/tmp,oh=u01/app/product/12.2.0/
    dbhome_1,sid=db122,target_home=/databases/product/19c/dbhome_1" -mode
    postfixups
    ```

18. Run `utlusts.sql`. This command verifies that all issues are fixed

    For example, in a CDB, where you use the `catcon` utility to run utlusts.sql, using the flag `-b` to set the log file base for the upgrade to `utlu21s`:

    ```
    $ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlu21s -d '''.'''
    utlusts.sql
    ```

    Because the command specifies `-b utlu21s`, upgrade results are stored in log file `utlu21s0.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.

19. (Conditional) For Oracle RAC environments only, enter the following commands to set the initialization parameter value for `CLUSTER_DATABASE` to `TRUE`:

```
ALTER SYSTEM SET CLUSTER_DATABASE=TRUE SCOPE=SPFILE;
```

20. Start Oracle Database, where *dbname* is the name of the database.

```
srvctl upgrade database -db db-unique-name -oraclehome oraclehome
srvctl enable database -db db-unique-name
```

21. Use SQL*Plus to shut down the database.

```
SHUTDOWN IMMEDIATE
```

22. Use `srvctl` to start up the database.

```
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.

Related Topics

- Managing Data Files and Temp Files

Upgrading Multitenant Architecture Sequentially Using Unplug-Plug

To upgrade pluggable databases (PDBs) that are in an earlier release multitenant container databases (CDBs), 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

- Backing Up CDBs and PDBs
- Unplugging a PDB from a CDB

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. To determine if the database is ready for upgrade, run AutoUpgrade with the `preupgrade` parameter, run in `analyze` mode. For example, with the database `salespdb` in the Oracle home `/u01/app/oracle/product/12.2.0/dbhome1`, checking for readiness to upgrade to Oracle Database 21c:
   a. Run `setenv ORACLE_HOME /u01/app/oracle/product/12.2.0/dbhome1`.
   b. Run `setenv ORACLE_SID salespdb`.
   c. Run `java -jar autoupgrade.jar -preupgrade "target_version=21,dir=/autoupgrade/test/log" -mode fixups`.
   d. Check `prefixups.xml` under the directory `/autoupgrade/test/log/salespdb/prefixups`.
2. Fix any issues AutoUpgrade detected that could not be fixed automatically.
3. Close the PDB you want to unplug.

   For example, use the following command to close the PDB `salespdb`:

   ```
   SQL> ALTER PLUGGABLE DATABASE salespdb CLOSE;
   ```

4. Log back in to `CDB$ROOT`:

   ```
   CONNECT / AS SYSDBA
   SQL> ALTER SESSION SET CONTAINER=CDB$ROOT;
   ```

5. 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
   ```

6. 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;
```

7. 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 run post-upgrade fixups, and to recompile the `INVALID` objects in the database, run the AutoUpgrade utility (`autoupgrade.jar`) using the option `-preupgrade` option with the mode `postfixups`:

   For example:

   ```
   java -jar autoupgrade.jar -preupgrade "dir=/
   tmp,inclusion_list=salespdb,target_home=/databases/product/19c/dbhome_1" -
   mode postfixups
   ```

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.

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

Upgrading Oracle Database Releases Using Replay Upgrade

To upgrade from an earlier release, you can use the Oracle Multitenant Replay Upgrade (Replay Upgrade) procedure to adopt a non-CDB to a PDB, or upgrade a PDB.

- Upgrading CDBs or PDBs Using Replay Upgrade
  You can upgrade an entire container database (CDB) and its pluggable databases (PDBs) using a Replay Upgrade, or you can upgrade individual PDBs.
- How to Disable or Enable Replay Upgrade
  By default, the Oracle Multitenant Replay Upgrade (Replay Upgrade) method is enabled for upgrades on PDBs and CDBs. However, you can enable or disable the use of the Replay Upgrade method.
- About Upgrading Non-CDBs to PDBs Using Replay Upgrade
  You can automate some of the steps to upgrade non-CDB Oracle Database software to the multitenant architecture by using the Oracle Multitenant Replay Upgrade (Replay Upgrade) method.
- Adopting and Upgrading a Non-CDB as a PDB with Replay Upgrade
  To simplify your upgrades, you can adopt (move) and upgrade a non-CDB into a PDB by using the Oracle Multitenant Replay Upgrade (Replay Upgrade) method.
- How the Replay Upgrade Procedure is Enabled or Disabled on CDBs and PDBs
  Learn how you can use the default Replay Upgrade, or choose to perform a classic script-based upgrade.
- Failure and Recovery Scenarios for Replay Upgrade Processes
  Learn how to check for errors and issues in log files and trace files for an Oracle Multitenant Replay Upgrade (Replay Upgrade).

Upgrading CDBs or PDBs Using Replay Upgrade

You can upgrade an entire container database (CDB) and its pluggable databases (PDBs) using a Replay Upgrade, or you can upgrade individual PDBs.

Before you start an upgrade, the following steps must be performed:

- Install the new release software for Oracle Database
- Prepare the new Oracle home
- Run AutoUpgrade with the `preupgrade` parameter.

> **Note:**
>
> When you plug in PDBs and upgrade the PDBs on PDB open using Replay Upgrade, Oracle recommends that you upgrade a number of PDBs equivalent to no more than one-fourth (¼) of the value of the Oracle Database initialization parameter `CPU_COUNT`, which specifies the number of CPU core (processors) available for Oracle Database to use. With traditional upgrades, the default number of PDBs upgraded at a time is no more than one-half (½) of the value of `CPU_COUNT`.

You can complete an upgrade using the Oracle Multitenant Replay Upgrade (Replay Upgrade) method either by using classic upgrade tools, or with the AutoUpgrade Utility.

Replay Upgrade Using Classic Upgrade Tools

To upgrade the entire CDB, including `CDB$ROOT` and all hosted PDBs, run the Parallel Upgrade Utility (`catctl.pl`) manually or implicitly, or use the `dbupgrade` script. The procedure is the same as in previous releases. However, starting with Oracle Database 21c, the upgrade utilities by default use the Replay Upgrade procedure. There is no change in command syntax.

Replay Upgrade Using Automatic Upgrade on PDB Plug-In To a New Release CDB

If you plug in an earlier release PDB to a new release CDB, then the CDB detects on opening the PDB that the PDB is an earlier release than the CDB, and automatically starts a Replay Upgrade process. Upgrade on PDB Open automatically upgrades the PDB using the Replay Upgrade synchronization feature. This optimization avoids opening the PDB in restricted mode, exposing the error in the `PDB_PLUG_IN_VIOLATIONS` view, so that you are not required to correct the error manually.

How to Disable or Enable Replay Upgrade

By default, the Oracle Multitenant Replay Upgrade (Replay Upgrade) method is enabled for upgrades on PDBs and CDBs. However, you can enable or disable the use of the Replay Upgrade method.

To disable the Parallel Upgrade Utility (`catctl.pl`) default of performing a Replay Upgrade, run the following command, on either `CDB$ROOT` or a particular PDB:

```
ALTER DATABASE UPGRADE SYNC OFF
```

To re-enable the Replay Upgrade behavior, enter the following command

```
ALTER DATABASE UPGRADE SYNC ON
```

You can also select a non-replay upgrade by setting the Parallel Upgrade Utility (`catctl.pl`) parameter `-t`, which forces a non-replay upgrade that uses the classic scripting method.

> **Note:**
>
> You can manage use of the Replay Upgrade method on the entire CDB, or on individual PDBs, depending on whether you are connected to `CDB$ROOT`, or to a particular PDB:
>
> - If `UPGRADE SYNC` is set to `OFF` in `CDB$ROOT`, then the Replay Upgrade method is not used for any PDBs plugged into the CDB.
> - If `UPGRADE SYNC` is set to `ON` in `CDB$ROOT`, but set to `OFF` for a PDB, then the Replay Upgrade method is not used for the PDB where `UPGRADE SYNC` is `OFF`, but the Replay Upgrade method is used for all other PDBs plugged into the CDB.
> - If `UPGRADE SYNC` is set to `ON` in `CDB$ROOT`, and set to `ON` for all PDBs (the default), then the Replay Upgrade method is used for all PDBs plugged into the CDB.

About Upgrading Non-CDBs to PDBs Using Replay Upgrade

You can automate some of the steps to upgrade non-CDB Oracle Database software to the multitenant architecture by using the Oracle Multitenant Replay Upgrade (Replay Upgrade) method.

The Replay Upgrade method is enabled by default for upgrades from earlier Oracle Database releases that are supported for direct upgrade to this Oracle Database release. The Replay Upgrade process is different from the classic method of running scripts, such as

noncdb_to_pdb.sql. For non-CDBs, after you describe the non-CDB by running DBMS_PDB.DESCRIBE, you plug in the non-CDB in to the new Oracle Database CDB. The Replay Upgrade method for upgrade is completed in two steps:

1. The non-CDB database is upgraded to the new Oracle Database release.
2. The non-CDB data dictionary is converted to a PDB data dictionary

Both of these steps are triggered when you run ALTER PLUGGABLE DATABASE OPEN. Both steps automatically replay SQL statements stored in the dictionary and complete the task of adopting the non-CDB to a PDB, and upgrading the database to the new release.

The benefit of using the Replay Upgrade method is to greatly simplify the upgrade workflow that you need to perform for PDB upgrades and conversions. The implicit non-CDB to PDB conversion simplifies the process of adopting and upgrading both non-CDB and PDB Oracle Database releases earlier than Oracle Database 21c to PDBs in a new release CDB.

Adopting and Upgrading a Non-CDB as a PDB with Replay Upgrade

To simplify your upgrades, you can adopt (move) and upgrade a non-CDB into a PDB by using the Oracle Multitenant Replay Upgrade (Replay Upgrade) method.

Before you start an upgrade, the following steps must be performed:

- Install the new release software for Oracle Database
- Prepare the new Oracle home
- Run AutoUpgrade with the preupgrade parameter.

To adopt a non-CDB as a PDB using the DBMS_PDB package and the Replay Upgrade method, complete the following procedure.

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 database description 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. To see why it is not compatible, check the view `PDB_PLUG_IN_VIOLATIONS`. Before you continue, you must correct all violations. For example, any version or patch mismatches should be resolved by running an upgrade, or running 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 a later 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. To enable the database open command to perform the Replay Upgrade, enter `ALTER DATABASE PROPERTY SET UPGRADE_PDB_ON_OPEN='true'`. For example:

   ```
   ALTER DATABASE PROPERTY SET UPGRADE_PDB_ON_OPEN='true';
   ```

9. Connect to the new PDB, and open it. At the time that the PDB is opened, the database is upgraded, and the non-CDB database data dictionary is converted to a PDB. For example:

```
ALTER PLUGGABLE DATABASE OPEN
```

You must open the new PDB for Oracle Database to complete upgrading the database. An error is returned if you attempt to open the PDB in read-only mode. When the PDB is opened, the non-CDB is adopted to a PDB, the data dictionary is converted, and the new PDB is integrated into the CDB. Messages from the Replay Upgrade are placed in the trace directory. After the PDB is opened, and the REPLAY UPGRADE is completed, its status is `NORMAL`.

To check the status of the upgrade, you can query the following views:

- To check for Replay Upgrade errors, use the view `DBA_REPLAY_UPGRADE_ERRORS`
- To check completeness, use `DBA_APPLICATIONS`. Check the `app_version` value for `app_name 'APP$CDB$CATALOG'` value. This value should be the new version of the PDB.
- Check the view `DBA_APP_ERRORS` for statement errors. This view lists the error message and statement text (`app_statement`) for any errors. In a successful upgrade, this view should not contain any rows for `app_name='APP$CDB$CATALOG'`.

10. Run datapatch to patch the new PDB:

```
datapatch -pdbs ncdb
```

11. Run postupgrade fixups:

```
java -jar autoupgrade.jar -preupgrade "target_home=/u01/app/oracle/product/
21.0.0/dbhome_1,dir=/autoupgrade/test/log,inclusion=ncdb" -mode postfixups
```

12. 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 might be in an `UNUSABLE` state. You can check the state of a PDB by querying the views `CDB_PDBS` or `DBA_PDBS`. You can learn more about PDB creation errors by checking the alert log. An unusable PDB can only be dropped. If the PDB is unusable, then it must be dropped before a PDB with the same name as the unusable PDB can be created.

How the Replay Upgrade Procedure is Enabled or Disabled on CDBs and PDBs

Learn how you can use the default Replay Upgrade, or choose to perform a classic script-based upgrade.

By default, when you run the Parallel Upgrade Utility (`catctl.pl`) manually, a Replay Upgrade is performed. When a PDB or a non-CDB is plugged into a new release container database (CDB). the PDB open detects whether the PDB or non-CDB requires an upgrade to be

compatible with the CDB. If you have opened the database, and for non-CDBs, run the `DBMS_PDB.DESCRIBE` procedure to construct an XML file that describes the non-CDB, then the upgrade automatically occurs when the PDB is opened. For example, using the Replay Upgrade method, you can relocate an earlier release PDB from an earlier release CDB to an Oracle Database 21c CDB PDB on plugging in the PDB, without needing to run the Parallel Upgrade Utility script `catctl.pl`, or using AutoUpgrade, Database Upgrade Assistant (DBUA), or using the `dbupgrade` command.

To perform a classic script-based upgrade, you can run `catctl.pl` with the option `-t`. When you plug in a non-CDB, you can also turn off or turn on the Replay Upgrade process by running `ALTER DATABASE UPGRADE SYNC OFF;` to turn Replay Upgrade off, or `ALTER DATABASE UPGRADE SYNC ON;` to turn Replay Upgrade on. This syntax modifies the `'PDB_UPGRADE_SYNC'` property, which is a number that is set to `2` for `ON` and `0` for `OFF`. If the number is positive, then `catctl.pl` uses the Replay Upgrade process. This property can be set in `CDB$ROOT`, or in the PDB. When disabled in `ROOT`, `catctl.pl` does not use the Replay Upgrade process for any PDBs. When enabled in `ROOT`, the PDB can either inherit this value, or set its own value.

Failure and Recovery Scenarios for Replay Upgrade Processes

Learn how to check for errors and issues in log files and trace files for an Oracle Multitenant Replay Upgrade (Replay Upgrade).

If a Replay Upgrade fails, then the `PDB_UPGRADE_SYNC` property is decremented by 1 for the PDB. If the Replay Upgrade fails twice, then `catctl.pl` falls back to using the classic script-based Parallel Upgrade Procedure method for completing the upgrade.

To determine the cause of Replay Upgrade errors, review the upgrade logs for statements that encounter errors. After a Replay Upgrade procedure runs, whether it is successful or unsuccessful, the query `DBA_APP_ERRORS` is run. Review the results of that query to see statement text and error messages for any statements that encounter errors. To see specific errors in the logs, you can also query the upgrade logs by using a `grep` command to locate any text strings of `'^ORA-'`.

Another diagnostic option is to review the trace files (`.trc`) . The trace files contain output for each statement run during the Replay Upgrade procedure. The files show statement text, whether statements succeeded or failed, error messages (if applicable), and the amount of time elapsed during the process. Look for lines that contain the prefix string `Replay Upgrade, PDB` *`ID`* where the variable *`ID`* is the PDB ID. For example: `Replay Upgrade, PDB 15`.

Manual Non-CDB Oracle Database Release Upgrades to Multitenant Architecture

To manage your non-CDB Oracle Database upgrade manually by using scripts, learn about upgrade scenarios and procedures.

Starting with Oracle Database 21c, non-CDB architecture is desupported. You must upgrade a non-CDB Oracle Database to a PDB on a CDB. You have two manual upgrade options available:

- Plug in the non-CDB Oracle Database to an Oracle Database 21c container database (CDB), and open the PDB in read-write, non-restricted mode. When the PDB is opened, the database is upgraded, and the data dictionary is converted from a non-CDB to a PDB.

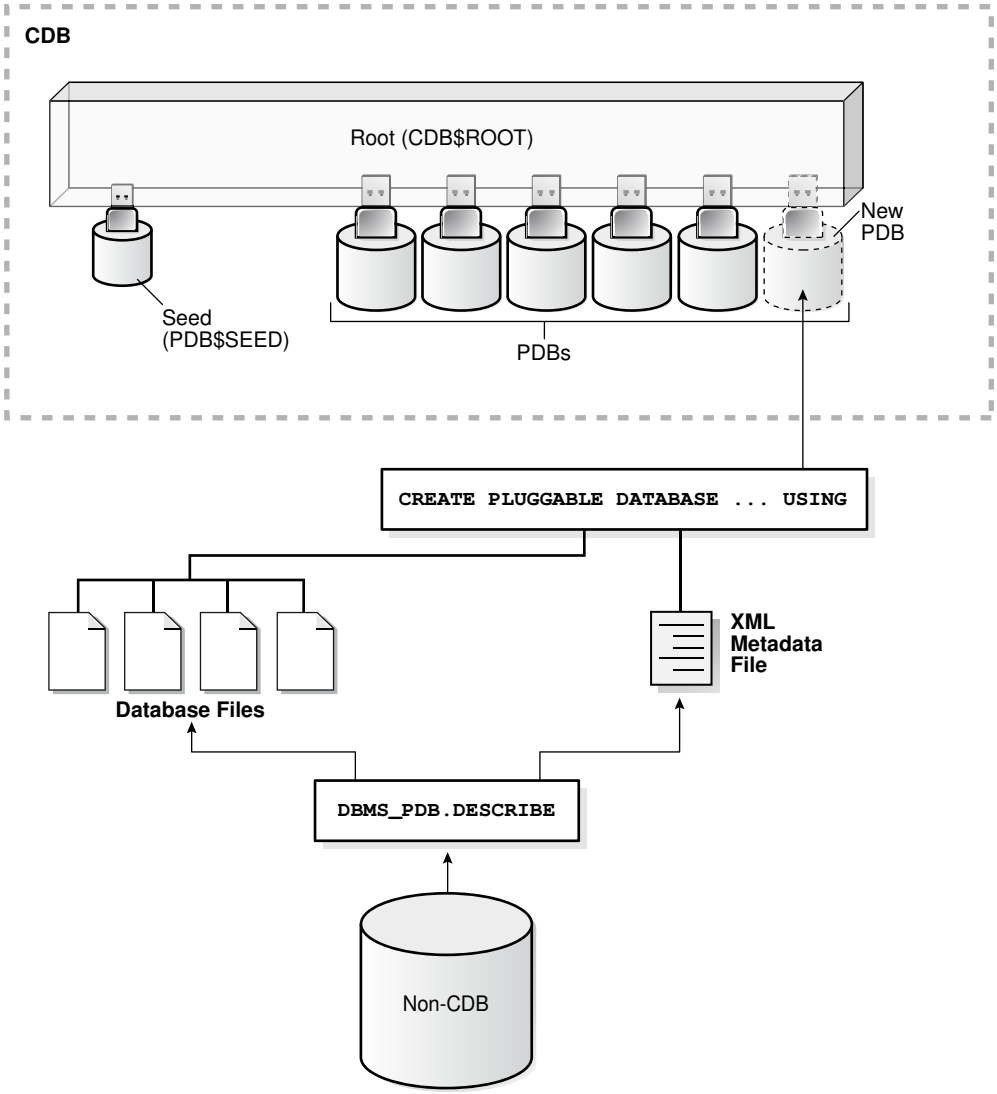
- Plug in the non-CDB Oracle Database to a same-release Oracle Database CDB, and convert the data dictionary from a non-CDB to a PDB. Then, upgrade the CDB and PDBs to Oracle Database 21c.
- About Adopting a Non-CDB as a PDB Using a PDB Plugin
  To manually adopt a non-CDB as a PDB, you generate an XML 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.
- Plugging In an Unplugged PDB
  You can create a PDB by plugging an unplugged PDB into a CDB.
- Manually Upgrading Non-CDB Architecture Oracle Databases
  Use this procedure after you have installed a CDB to upgrade an earlier release non-CDB architecture Oracle Database, making it a PDB, and plugging the PDB into a CDB.

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 5-1 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.

> **Note:**
>
> A multitenant container database is the only supported architecture in Oracle Database 21c and later releases. While the documentation is being revised, legacy terminology may persist. In most cases, "database" and "non-CDB" refer to a CDB or PDB, depending on context. In some contexts, such as upgrades, "non-CDB" refers to a non-CDB from a previous release.

In a multitenant environment, pluggable databases (PDBs) can be plugged in and out of a multitenant container database (CDB) or an application container.

- `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*

Plugging In an Unplugged PDB

You can create a PDB by plugging an unplugged PDB into a CDB.

Manually Upgrading Non-CDB Architecture Oracle Databases

Use this procedure after you have installed a CDB to upgrade an earlier release non-CDB architecture Oracle Database, making it a PDB, and plugging the PDB into a CDB.

> **Note:**
>
> Starting with Oracle Database 21c, non-CDB architecture is desupported. You must upgrade a non-CDB Oracle Database to a PDB on a CDB.

Before using this procedure, complete the following steps:

- Install the new release Oracle Database software
- Prepare the new multitenant architecture Oracle home
- Run AutoUpgrade with the preupgrade parameter, to check your source database system's readiness for upgrade to the new release.

Steps:

1. If you have not done so, run AutoUpgrade using the preupgrade parameter. Review the output, and correct all issues noted in the output before proceeding.
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`.

      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\21.0.0\DB, then enter the following command:

```
C:\> ORADIM -NEW -SID ORCL -SYSPWD TWxy5791 -MAXUSERS 10
-STARTMODE AUTO -PFILE C:\ORACLE\PRODUCT\21.0.0\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 container database.

> **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.

The following table describes the variables for using ORADIM when upgrading manually:

Table 5-2 ORADIM Variables and Functions

ORADIM Variable	Description
-SID *sid*	The same SID name as the SID for the database that you are upgrading
-SYSPWD *password*	The SYS password for the upgraded Oracle Database instance. This is the password for the user connected with SYSDBA privileges. Default Oracle Database Security settings require that passwords must be at least eight characters. You are not permitted to use passwords such as welcome and oracle.
- MAXUSERS *value*	The maximum number of user accounts that can be granted SYSDBA or SYSOPER privileges.
-PFILE *oracle-home-path*	The location of the parameter file (PFILE) in the Oracle home location for the upgraded Oracle Database release. Ensure that you specify the full path name with the -PFILE option, including the drive letter of the Oracle home location.

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. Copy the SPFILE.ORA or INIT.ORA file from the old Oracle home to the new Oracle home.
10. 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/21.0.0/dbhome_1/rdbms/admin
$ ./sqlplus
```

11. Connect to the database that you want to upgrade using an account with SYSDBA privileges:

```
Enter user-hame: 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
    ```

    Otherwise, run `$ORACLE_HOME/bin/dbupgrade`.

    > **Note:**
    >
    > - When you run the Parallel Upgrade Utility command, 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 the AutoUpgrade utility (`autoupgrade.jar`) with the option `-preupgrade` using the mode `postfixups`.

    For example:

    ```
    java -jar autoupgrade.jar -preupgrade -mode postfixups
    ```

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 or upgraded Oracle Database release, and using the command `rhpctl upgrade database` to upgrade the earlier release Oracle Database in a fleet image deployment. 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.

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 Infrastructure 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 uses 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

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.

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 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

With the Resume option for Parallel Upgrade Utility, 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 21c, upgrades to non-CDB architecture are desupported.

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 utlu21s -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 utlu21s -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)
  If you want to rerun upgrades on Oracle Database using multitenant database architecture, then you have four options.
- 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 postupgrade tasks, and consider recommendations for the new release.

Options for Rerunning the Upgrade for Multitenant Databases (CDBs)

If you want to rerun upgrades on Oracle Database using multitenant database architecture, then you have four options.

- 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 pluggable database (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 5-13 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 5-14 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 5-15 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 5-16 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 pluggable database (PDB) by using an inclusion list.

You can use an inclusion list to specify a list of container databases (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 5-17 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'
   ```

6

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.

- Error Upgrading Non-CDB Oracle Databases
  If you attempt to upgrade a non-CDB Oracle Database release, you receive the error `ORA-01722: invalid number`.
- Fixed View Queries Restriction When 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.
- Resolving PDBs in Restricted Mode After Successful Upgrades
  If your upgrade is successful, but the upgraded PDBs are in Restricted Mode, then this may be due to components set to `OPTION OFF`.
- 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.
- Fixing Failures to Complete Registration of Listeners with DBUA
  On the Database Upgrade Assistant Progress step, a window appears with this warning: "Unable to create database entry in the directory service. No Listeners configured."

Related Topics

- Rerunning Upgrades for Oracle Database
  Use these options to rerun upgrades.

Error Upgrading Non-CDB Oracle Databases

If you attempt to upgrade a non-CDB Oracle Database release, you receive the error `ORA-01722: invalid number`.

Starting with Oracle Database 21c, you must use the multitenant architecture for Oracle Database upgrades. When you attempt to upgrade a non-CDB Oracle Database release to Oracle Database 21c, and do not upgrade to a multitenant architecture, you receive the following error:

```
SELECT TO_NUMBER('UPGRADE OF A NON-CDB TO TARGET RELEASE IS NOT SUPPORTED') *
ERROR at line 1: ORA-01722: invalid number
```

To resolve this issue, use one of the Non-CDB to CDB upgrade methods, so that you upgrade to a multitenant architecture.

Fixed View Queries Restriction When 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

Resolving PDBs in Restricted Mode After Successful Upgrades

If your upgrade is successful, but the upgraded PDBs are in Restricted Mode, then this may be due to components set to `OPTION OFF`.

If a PDB is opened in restricted mode, then query `pdb_plug_in_violations` for errors to see if this result is due to one or more components in that PDB with `OPTION OFF` status. If the result of your query shows a database option mismatch for the same components, then close and restart the PDB. After the restart, check to see of the PDB status for `RESTRICTED` mode is changed from `YES` to `NO`.

For example, look for `Database option mismatch` results similar to the following:

```
SQL> select time, message from pdb_plug_in_violations where status='PENDING'
and type='ERROR';
```

```
TIME ----------------MESSAGE-----------------------------------------

11-SEP-20 12.40.56.096230 PM
Database option APS mismatch: PDB installed version 21.0.0.0.0. CDB installed
version NULL.

11-SEP-20 12.40.56.096789 PM
Database option DV mismatch: PDB installed version 21.0.0.0.0. CDB installed
version NULL.

11-SEP-20 12.40.56.097315 PM

Database option OLS mismatch: PDB installed version 21.0.0.0.0. CDB installed
version NULL.

11-SEP-20 12.40.56.098157 PM
Database option XOQ mismatch: PDB installed version 21.0.0.0.0. CDB installed
version NULL.
```

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

- Pre-Upgrade Information Tool and AutoUpgrade Preupgrade

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 did not run AutoUpgrade with the `preupgrade` parameter before starting the upgrade, then the `catctl.pl` and `catupgrd.sql` scripts terminate with errors such as the following:

```
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 use this procedure to correct the problem:

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 AutoUpgrade utility using the preupgrade parameter, and correct the issues that it reports in the `upgrade.xml` file.

Related Topics

- Pre-Upgrade Information Tool for Oracle Database
  Learn how to obtain the same features previously offered through the Pre-Upgrade information tool (`preupgrade.jar`) by using AutoUpgrade with the `preupgrade` clause.

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, the upgrade utility script `utlusts.sql` displays an upgrade report.

Example of an Upgrade Summary Report

```
Oracle Database Release 20 Post-Upgrade Status Tool    12-04-2019 08:18:1
Container Database: AIME1
[CON_ID: 1 => CDB$ROOT]

Component                                Current         Full     Elapsed Time
Name                                     Status          Version  HH:MM:SS

Oracle Server                           UPGRADED      20.1.0.0.0  00:22:08
JServer JAVA Virtual Machine            UPGRADED      20.1.0.0.0  00:09:35
Oracle XDK                              UPGRADED      20.1.0.0.0  00:01:36
Oracle Database Java Packages           UPGRADED      20.1.0.0.0  00:00:14
OLAP Analytic Workspace                 UPGRADED      20.1.0.0.0  00:00:44
Oracle Label Security                   UPGRADED      20.1.0.0.0  00:00:17
Oracle Database Vault                   UPGRADED      20.1.0.0.0  00:01:20
Oracle Text                             UPGRADED      20.1.0.0.0  00:01:21
Oracle Workspace Manager                UPGRADED      20.1.0.0.0  00:01:20
Oracle Real Application Clusters        UPGRADED      20.1.0.0.0  00:00:06
Oracle XML Database                     UPGRADED      20.1.0.0.0  00:03:08
Oracle Multimedia                       UPGRADED      20.1.0.0.0  00:01:33
Spatial                                 UPGRADED      20.1.0.0.0  00:07:51
Oracle OLAP API                         UPGRADED      20.1.0.0.0  00:00:12
Oracle Locator                          UPGRADED      20.1.0.0.0  00:00:00
Datapatch                                                         00:00:23
Final Actions                                                     00:00:43
Post Upgrade                                                      00:00:17

Total Upgrade Time: 00:48:56 [CON_ID: 1 => CDB$ROOT]

Database time zone version is 31. It is older than current release time
zone version 34. Time zone upgrade is needed using the DBMS_DST package.


Oracle Database Release 20 Post-Upgrade Status Tool    12-04-2019 09:12:4
Container Database: AIME1
[CON_ID: 3 => CDB1_PDB1]

Component                                Current         Full     Elapsed Time
Name                                     Status          Version  HH:MM:SS
```

```
Oracle Server                               UPGRADED      20.1.0.0.0  00:31:00
JServer JAVA Virtual Machine                UPGRADED      20.1.0.0.0  00:04:25
Oracle XDK                                  UPGRADED      20.1.0.0.0  00:01:38
Oracle Database Java Packages               UPGRADED      20.1.0.0.0  00:00:13
OLAP Analytic Workspace                     UPGRADED      20.1.0.0.0  00:01:03
Oracle Label Security                       UPGRADED      20.1.0.0.0  00:00:11
Oracle Database Vault                       UPGRADED      20.1.0.0.0  00:01:38
Oracle Text                                 UPGRADED      20.1.0.0.0  00:00:34
Oracle Workspace Manager                    UPGRADED      20.1.0.0.0  00:00:53
Oracle Real Application Clusters            UPGRADED      20.1.0.0.0  00:00:00
Oracle XML Database                         UPGRADED      20.1.0.0.0  00:03:04
Oracle Multimedia                           UPGRADED      20.1.0.0.0  00:01:15
Spatial                                     UPGRADED      20.1.0.0.0  00:06:48
Oracle OLAP API                             UPGRADED      20.1.0.0.0  00:00:18
Oracle Locator                              UPGRADED      20.1.0.0.0  00:00:00
Datapatch                                                               00:00:23
Final Actions                                                           00:00:44
Post Upgrade                                                            00:00:20

Total Upgrade Time: 00:51:12 [CON_ID: 3 => CDB1_PDB1]

Database time zone version is 31. It is older than current release time
zone version 34. Time zone upgrade is needed using the DBMS_DST package.


Oracle Database Release 20 Post-Upgrade Status Tool    12-04-2019 09:20:0
Container Database: AIME1
[CON_ID: 2 => PDB$SEED]

Component                               Current         Full     Elapsed Time
Name                                    Status          Version  HH:MM:SS

Oracle Server                             VALID      20.1.0.0.0  00:30:56
JServer JAVA Virtual Machine              VALID      20.1.0.0.0  00:04:29
Oracle XDK                                VALID      20.1.0.0.0  00:01:38
Oracle Database Java Packages             VALID      20.1.0.0.0  00:00:12
OLAP Analytic Workspace                   VALID      20.1.0.0.0  00:00:55
Oracle Label Security                     VALID      20.1.0.0.0  00:00:14
Oracle Database Vault                     VALID      20.1.0.0.0  00:01:41
Oracle Text                               VALID      20.1.0.0.0  00:00:40
Oracle Workspace Manager                  VALID      20.1.0.0.0  00:00:53
Oracle Real Application Clusters     OPTION OFF      20.1.0.0.0  00:00:00
Oracle XML Database                       VALID      20.1.0.0.0  00:03:04
Oracle Multimedia                         VALID      20.1.0.0.0  00:01:15
Spatial                                   VALID      20.1.0.0.0  00:06:49
Oracle OLAP API                           VALID      20.1.0.0.0  00:00:19
Oracle Locator                            VALID      20.1.0.0.0  00:00:00
Datapatch                                                         00:00:25
Final Actions                                                     00:00:45
Post Upgrade                                                      00:00:21
Post Compile                                                      00:07:11

Total Upgrade Time: 00:58:24 [CON_ID: 2 => PDB$SEED * ]
Asterisks denotes compilation time has been included during the upgrade
process.
```

```
Database time zone version is 31. It is older than current release time
zone version 34. Time zone upgrade is needed using the DBMS_DST package.


Upgrade Times Sorted In Descending Order

Total Upgrade Time: 00:58:24 [CON_ID: 2 => PDB$SEED * ]
Total Upgrade Time: 00:51:12 [CON_ID: 3 => CDB1_PDB1]
Total Upgrade Time: 00:48:56 [CON_ID: 1 => CDB$ROOT]
Grand Total Upgrade Time:    [0d:1h:53m:17s]
```

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
   ```

Fixing Failures to Complete Registration of Listeners with DBUA

On the Database Upgrade Assistant Progress step, a window appears with this warning: "Unable to create database entry in the directory service. No Listeners configured."

In this scenario, you have completed the following steps:

1. Created a listener in your earlier release Oracle home using Net Configuration Assistant (NetCA) or Net Manager.
2. Installed the new Oracle Database release in the new Oracle home, and registered it against the listener.
3. You want to register the listener to Oracle Internet Directory (OID), and upgrade the database, or you want to register the listener on OID on the new Oracle home.

> **Note:**
>
> If the listener is running from another Oracle home on the server, or the listener is running from the current Oracle home, but it is not configured in the `LISTENER.ORA` file (that is, it uses an automatically generated default configuration), then DBCA is unable to locate the listener.

DBUA is no longer performing direct registration of the upgraded database to OID during the upgrade itself.

To resolve this issue, complete the following tasks:

1. De-register the listener from OID.
2. Perform the database upgrade.
3. Register again the migrated listener to the OID registry on the new release Oracle Database Oracle home.

7

Postupgrade Tasks for Oracle Database

After you upgrade Oracle Database, complete required postupgrade tasks, and consider recommendations for the new release.

> **Note:**
>
> If you use AutoUpgrade to complete the upgrade, then many of these tasks may be done automatically as part of the upgrade.

- 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
  To check the state of the Oracle Data Dictionary for diagnosing upgrades and migrations, use one of three methods.
- 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 types of upgrades.
- 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.

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 located in the path `$ORACLE_HOME/rdbms/admin`. The 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

To check the state of the Oracle Data Dictionary for diagnosing upgrades and migrations, use one of three methods.

Example 7-1 Run a SQL Query on DBA_REGISTRY

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_full,status
from dba_registry order by comp_id;
```

Example 7-2 Run SQL Queries to Check for Invalid Objects

To check for invalid objects, you can perform SQL queries, similar to the following examples:

1. This query list all the invalid objects in the database:

   ```
   SQL> select owner, object_name, object_type from dba_invalid_objects order
   by owner, object_type, object_name;
   ```

2. After you have upgraded the database, and you have run `utlrp.sql`, Oracle-maintained objects should be valid.

   To check to ensure Oracle-maintained objects are valid, enter the following query:

   ```
   SQL> select owner, object_name, object_type from sys.dba_invalid_objects
   where oracle_maintained='Y';
   ```

3. To list any invalid application objects in the database, enter the following query:

   ```
   SQL> select owner, object_name, object_type from sys.dba_invalid_objects
   where oracle_maintained='N';
   ```

Example 7-3 Run the `dbupgdiag.sql` Script

(Optional) If you want to obtain further information about the upgrade, you can choose to run the `dbupgdiag.sql` script. The `dbupgdiag.sql` script collects diagnostic information about the status of the database, either before or after the upgrade. Download the script from My Oracle Support note 556610, and run the script as the database `SYS` user. The script generates the diagnostic information in a readable format, in a log file with the name file `db_upg_diag_`*`sid`*`_`*`timestamp`*`.log`, where *`sid`* is the Oracle system identifier for the database, and *`timestamp`* is the time that the file is generated.

For example, where you download and place the script in the directory `/u01/dbupgdiag-script`:

```
/u01/dbupdiag-script/ $ sqlplus / as sysdba
sql> alter session set nls_language='American';
```

```
sql> @dbupgdiag.sql
sql> exit
```

You can run the script in SQL*Plus both before the upgrade on the source database, and after the upgrade on the upgraded database. For more information about the script, refer to the instructions and the output example file in My Oracle Support Note 556610.1.

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
  Check that required operating system environment variables point to the directories of the new Oracle Database release.
- Recompile Invalid Objects in the Database
  After you install, patch, or upgrade a database, recompile invalid objects on the CDB and PDBs with a recompilation driver script.
- Track Invalid Object Recompilation Progress
  Use these SQL queries to track the progress of `utlrp.sql` script recompilation of invalid objects.
- Update Listener Files Location on Oracle RAC Cluster Member Upgrades
  If you do not use a shared Oracle Base Home, and you perform an upgrade on an Oracle Database instance that is an Oracle Real Application Clusters member node, then you must update your listener path.
- 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 12c Release 2 (12.2) and later releases, you must manually upgrade user tables that depend on Oracle-Maintained types.
- 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.
- Drop Earlier Release Oracle APEX
  To avoid invalid objects, drop Oracle APEX releases earlier than Application Express (APEX) 19.1.
- 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 reenable Oracle Database Vault, or revoke Oracle Database Vault role granted for 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

Check that required operating system environment variables point to the directories of the new Oracle Database release.

Typically, operating system environment variables are set in profiles and shell scripts. Confirm that the following Oracle user environment variables point to the directories of the new Oracle home:

- `ORACLE_HOME`
- `PATH`

Look for other environment variables that refer to the earlier release Oracle home, such as `LD_LIBRARY_PATH`. In general, you should replace all occurrences of the old Oracle home in your environment variables with the new Oracle home paths.

Related Topics

- Step 2: Ensure That the Required Environment Variables Are Set

Recompile Invalid Objects in the Database

After you install, patch, or upgrade a database, recompile invalid objects on the CDB and PDBs with a recompilation driver script.

By default, AutoUpgrade performs a recompilation of invalid Oracle objects, which is controlled by the configuraiton file run_utlrp local parameter (default: *`prefix`*`.run_utlrp=yes`). In addition, Oracle provides the recompilation scripts `utlrp.sql` and `utlprp.sql`. These scripts are located in the *`Oracle_home`*`/rdbms/admin` directory.

> **Note:**
>
> Starting with AutoUpgrade 23.1, when you run the AutoUpgrade utility, AutoUpgrade runs the `utlprpom.sql` script, and does not run `utlrp.sql`. When AutoUpgrade is used for upgrades to Oracle Database 12c Release 2 (12.2.0.1) and later releases, AutoUpgrade only recompiles invalid objects owned by Oracle-maintained schemas. Because database upgrades do not need to touch user objects, AutoUpgrade maintains this policy when it recompiles invalid objects.

After installing a database, recomplile all 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.

After patching or upgrading a database, there is more than one approach you can use to recompile invalid Oracle-owned and user-owned objects:

Recompile all invalid objects (the invalid objects in both Oracle and user schemas) by using utlrp.sql or utlprp.sql.

If time is a factor and the type of invalid objects is predominately application owned, then you can recompile Oracle-owned invalid objects first, and defer recompiling application-owned invalid objects to a later time. To recompile invalid objects in Oracle schemas, use utlprpom.sql. To recompile the remaining invalid objects, use utlrp.sql or utlprp.sql.

> **Note:**
>
> When you use either utlprp.sql or utlprpom.sql, note that both scripts require you to define the degree of parallelism that the script should use, or determine the number of parallel recompile jobs to use.

The script uses syntax as follows, where *base* is the base name you want to have given to log files, *N* is the number of PDBs on which you want to run recompilation jobs in parallel (degrees of parallelism), *script.sql* is the Oracle recompilation script you chose to use, and *P* is the number of PDBs on which you want to run in parallel:

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -b base -
d $ORACLE_HOME/rdbms/admin
          -n N -l /tmp script.sql '--pP'
```

Suppose you are running recompilation in a CDB using the log file base name recomp, with a degrees of parallelism setting of 3 jobs per PDB container, the script you choose to use is

`utlprp.sql`, and you want to recompile across at most 10 PDBs at a time. In that case, the syntax you use to run the recompile operation is similar to the following,

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -b recomp -
d $ORACLE_HOME/rdbms/admin -n 10 -l /tmp utlprp.sql '--p3'
```

Related Topics

- Syntax and Parameters for catcon.pl

Track Invalid Object Recompilation Progress

Use these SQL queries to track the progress of `utlrp.sql` script recompilation of invalid objects.

> **Note:**
>
> If you upgraded using the AutoUpgrade utility, then AutoUpgrade automatically takes care of this task during the upgrade. You do not need to perform this task.

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 7-4 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 7-5 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 7-6 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.

Update Listener Files Location on Oracle RAC Cluster Member Upgrades

If you do not use a shared Oracle Base Home, and you perform an upgrade on an Oracle Database instance that is an Oracle Real Application Clusters member node, then you must update your listener path.

If you do not use a shared Oracle Base Home, and you perform an upgrade on an Oracle Database instance that is an Oracle Real Application Clusters cluster member node, but you do not use Database Configuration Assistant (DBCA), then you must manually update all of the remote `tnsnames.ora` files. If the files are not updated, then the following error occurs:

```
TNS-03505: Failed to resolve name
```

Starting with Oracle Database 21c, the default network administration directory changes from the previous default in the local Oracle home, *`Oracle_home`*`/network` (for example, `/u01/app/oracle/product/19.1.0/dbhome_1/network`), to a new location. The new default location is the shared Oracle Base Home, in the path *`ORACLE_BASE`*`/homes/`*`HOME_NAME`*`/network/admin`. For example:`/u01/app/oracle/homes/OraDB20Home1/network/admin` is the Oracle home of a specific Oracle Real Application Clusters (Oracle RAC) instance, which is located in the default path for read-only Oracle homes. That file path is the default location of the local `tnsnames.ora` and `sqlnet.ora` files.

During the upgrade, Net Configuration Assistant (NetCA, or `netca`) updates the location of the network listener files, `tnsnames.ora` and `sqlnet.ora` on the local host. However, NetCA does not update the location of those network listener files for the instance on all cluster member nodes. To ensure that all cluster member nodes can resolve service requests to the upgraded cluster member node instance, you must do one of the following:

- On each cluster member node, Manually copy the `tnsnames.ora` and `sqlnet.ora` files from the existing location to *`ORACLE_BASE`*`/homes/`*`HOME_NAME`*`/network/admin/`.
- Set an environment variable on the upgraded cluster member node to point to the existing network administration files location.

p

To ensure that *WORKAROUND:----------cp tnsnames.ora and sqlnet.ora to $ORACLE_BASE/homes/OraDB20Home1/network/admin/sqlnet.oraorexport TNS_ADMIN=$ORACLE_HOME/network/admin*

Example 7-7 Copy TNSNAMES.ORA and SQLNET.ORA to New Default Network Administration Directory

In this example, you copy the existing `tnsnames.ora` and `sqlnet.ora` files to the new default network location on each cluster member node

1. Log in as the Oracle installation owner, and change directory to the earlier release network administration directory. For example:

   ```
   cd $ORACLE_HOME/network
   ```

2. Use secure copy (`scp`) to copy the listener files to the default network directory location on another cluster member node. For example, where `racnode2` is a cluster member node to which you want to copy the listener files:

```
cd $OLD_ORACLE_HOME/network
scp tnsnames.ora sqlnet.ora \
oracle@racnode2:/u01/app/oracle/homes/OraDB20Home1/network/admin/
```

Example 7-8 Set the TNS_ADMIN Environment Variable

On the upgraded node, log in as the Oracle user, and then set an environment variable for TNS_ADMIN to point to the location of your existing listener files. For example:

```
/home/oracle oracle> $ export TNS_ADMIN=$ORACLE_HOME/network/admin
```

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, if you do not use AutoUpgrade for your database upgrade, then 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. If you perform your upgrade using either AutoUpgrade or DBUA, then these utilities automatically point `oratab` to the new Oracle home. However, regardless of the method you use to upgrade, you must check client scripts.

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.

Related Topics

- Step 2: Set Operating System Environment Variables
- My Oracle Support Note 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, if you use AutoUpgrade, review the AutoUpgrade report on invalid objects. If you use a replay upgrade, then check to ensure that any packages that you may 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 12c Release 2 (12.2) and later releases, you must manually upgrade user tables that depend on Oracle-Maintained types.

> **Note:**
>
> If you upgraded using the AutoUpgrade utility, then AutoUpgrade automatically takes care of this task during the upgrade. You do not need to perform this task.

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
```

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

Drop Earlier Release Oracle APEX

To avoid invalid objects, drop Oracle APEX releases earlier than Application Express (APEX) 19.1.

The package `DBMS_OBFUSCATION_TOOLKIT` was deprecated in Oracle Database 11g Release 2 (11.2). Starting in Oracle Database 21c, `DBMS_OBFUSCATION_TOOLKIT` is desupported, and replaced with `DBMS_CRYPT`. In releases earlier than Oracle Database 21c, Oracle APEX continued to have a dependency on `DBMS_OBFUSCATION_TOOLKIT`. In Oracle Database 21c, which includes APEX release 19.2, this dependency is removed.

During Oracle Database upgrade, APEX is not automatically upgraded to the release shipped with the Oracle Database release. Before upgrade, when you run AutoUpgrade using the `preupgrade` parameter, the output `upgrade.xml` file reports that you must upgrade APEX releases earlier than APEX 19.1.0.00.15, because of the earlier release dependency on `DBMS_OBFUSCATION_TOOLKIT`.

To avoid `INVALID` objects, either before or after you upgrade Oracle Database, you must upgrade APEX to at least the version that ships with Oracle Database 21c, and then drop earlier releases of APEX.

1. Log in to the new Oracle Database Oracle home
2. Upgrade Oracle APEX, using at least Oracle Application Express 19.2, which is shipped in the Oracle home (APEX 19.2), or a later version, when it is available
3. Drop earlier release APEX users.
   For example:

   ```
   drop user APEX_050000 cascade;
   drop user APEX_040200 cascade;
   drop user APEX_030100 cascade;
   ```

4. Drop earlier release APEX SYS owned objects.
   For example:

   ```
   drop package SYS.WWV_DBMS_SQL;
   ```

> **Note:**
>
> Starting with Oracle APEX 18c (18.1), the `SYS.WWV_DBMS_SQL` object is appended with the Oracle APEX release schema.
> For example:
>
> ```
> SYS.WWV_DBMS_SQL_APEX_180100
> ```

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 7-9 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 reenable Oracle Database Vault, or revoke Oracle Database Vault role granted for upgrade.

- Upgrading Oracle Database Without Disabling Oracle Database Vault
  To upgrade to Oracle Database 12c Release 2 (12.2.0.1) or later releases, either grant the `DV_PATCH_ADMIN` role to `SYS` commonly in the root container, and revoke after the upgrade, or disable Oracle Database Vault and reenable it after upgrade.
- Postupgrade Scenarios with Oracle Database Vault
  Postupgrade tasks for Oracle Database Vault change, depending on your target Oracle Database release, and the option you chose to prepare for upgrade.

Upgrading Oracle Database Without Disabling Oracle Database Vault

To upgrade to Oracle Database 12c Release 2 (12.2.0.1) or later releases, either grant the `DV_PATCH_ADMIN` role to `SYS` commonly in the root container, and revoke after the upgrade, or disable Oracle Database Vault and reenable it after upgrade.

If Oracle Database Vault is enabled and you are upgrading an entire CDB, then use one of the following methods:

- CDB upgrade method 1: Temporarily grant the `DV_PATCH_ADMIN` to user `SYS` commonly by logging into the root container as a common user with the `DV_OWNER` role, and then issuing the `GRANT DV_PATCH_ADMIN TO SYS CONTAINER=ALL` statement. Oracle Database Vault controls will be in the same state as it was before the upgrade. When the upgrade is complete, log into the root container as the `DV_OWNER` user, and revoke the `DV_PATCH_ADMIN` role from `SYS` by issuing the `REVOKE DV_PATCH_ADMIN FROM SYS CONTAINER=ALL` statement.
- CDB upgrade method 2: Log into each container as a user who has the `DV_OWNER` role, and then run the `DBMS_MACADM.DISABLE_DV` procedure. You must first disable Oracle Database Vault on the PDBs, and then after that, disable Oracle Database Vault on the root container last. If you are upgrading only one PDB, then you can disable Oracle Database Vault in that PDB only. After you have completed the upgrade, you can enable Oracle Database Vault by logging into each container as the `DV_OWNER` user and then executing the `DVSYS.DBMS_MACADM.ENABLE_DV` procedure. The order of enabling Oracle Database Vault must be the root container first and PDBs afterward. You can enable the PDBs in any order, but the root container must be enabled first.

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.

> **Note:**
>
> This procedure applies to non-CDB upgrades as well

Related Topics

- Oracle Database - Overview of Database Patch Delivery Methods for 12.2.0.1 and greater (Doc ID 2337415.1)
- Disabling and Enabling Oracle Database Vault *Oracle Database Vault Administrator's Guide*

Postupgrade Scenarios with Oracle Database Vault

Postupgrade tasks for Oracle Database Vault change, depending on your target Oracle Database release, and the option you chose to prepare for upgrade.

Upgrades to Oracle Database 21c and Later

You must choose one of the following options:

- Grant the `DV_PATCH_ADMIN` role to `SYS` commonly (`container=all`).

- Disable Oracle Database Vault before upgrade.

If you granted the `DV_PATCH_ADMIN` role to `SYS` before the upgrade, then revoke the `DV_PATCH_ADMIN` role from `SYS` after the upgrade. If you disabled Oracle Database Vault, then reenable it after the upgrade is complete.

Upgrades to Oracle Database 18c and 19c

You do not need to disable Oracle Database Vault.

> **Note:**
>
> For all upgrades, after the upgrade is complete, Oracle Database Vault has the same enforcement status that was in place for your source database 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 types of upgrades.

- 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.
- Run AutoUpgrade Postupgrade Checks
  If you did not run AutoUpgrade in `deploy` mode, then run Autoupgrade with the `preupgrade` parameter, run in `postfixups` mode.
- Gathering Dictionary Statistics After Upgrading
  To help to assure good performance, use this procedure to gather dictionary statistics after completing your upgrade.
- 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`.
- Regathering Fixed Objects Statistics with DBMS_STATS
  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.
- 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.
- 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`.
- Turn off Traditional Auditing in Upgraded Oracle Databases
  Traditional auditing is deprecated in Oracle Database 21c, and is desupported in Oracle Database 23c. Oracle recommends that you turn off traditional audit in your database and use only 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 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 the Time Zone File Version After Upgrading Oracle Database
  If the AutoUpgrade preupgrade report instructs you to upgrade the time zone files after completing the database upgrade, and you do not set AutoUpgrade to complete this task for you, then use any of the supported methods to upgrade the time zone file.
- Enabling Disabled Release Update Bug Fixes in the Upgraded Database
  Because bug fixes in Release Updates that can cause execution plan changes are disabled, Oracle recommends that you enable the disabled bug fixes that you want to use.
- 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.

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

Run AutoUpgrade Postupgrade Checks

If you did not run AutoUpgrade in `deploy` mode, then run Autoupgrade with the `preupgrade` parameter, run in `postfixups` mode.

> **Note:**
>
> If you ran AutoUpgrade in `deploy` mode, then this step was already completed for you, so you do not need to complete it now.

To see how to check your database after upgrades, use the following example.

Example 7-10 Running AutoUpgrade Using Postupgrade Fixup Mode

1. Set the Oracle home environment to the source Oracle Database home:

   ```
   setenv ORACLE_HOME /u01/app/oracle/product/12.2.0/dbhome_1
   ```

   .

2. Set the Oracle System Identifier (SID) to the source Oracle Database SID:

   ```
   setenv ORACLE_SID db122
   ```

   .

3. Run AutoUpgrade using the preupgrade parameter in postfixups mode, setting the target home to the target Oracle Database Oracle home. For example:

   ```
   java -jar autoupgrade.jar -preupgrade "target_home=/u01/app/oracle/product/
   21.0.0/dbhome_1,dir=/autoupgrade/test/log" -mode postfixups
   ```

4. Check the results of the postfixup script checks in the file `postfixups.xml` under directory `/autoupgrade/test/log/db122/102/postfixups`.

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 12c 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.

> **Note:**
>
> If you completed your upgrade using the AutoUpgrade utility, then you do not need to complete this task. The AutoUpgrade utility completes it for you.

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*

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

Regathering Fixed Objects Statistics with DBMS_STATS

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:**
>
> To provide the most correct fixed object statistics for performance tuning, Oracle strongly recommends that you gather baseline statistics at a point when the system is running with a representative workload. For useful results, never run `DBMS_STATS.GATHER_FIXED_OBJECTS_STATS` immediately after the upgrade.

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;
```

Related Topics

- Gathering Database Statistics

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. In Oracle Database 21c and later release, the IGNORECASE parameter for the orapwd file is desupported. All newly created password files are case-sensitive. 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.

Upgraded password files from earlier Oracle Database releases can retain original case-insensitive passwords. To ensure that password files are case-sensitive, Oracle recommends that you force case sensitivity by migrating password files from one format to another, using the following syntax:

```
orapwd input_file=input_password _file file=output_password_file
```

To secure your database, create passwords in a secure fashion. If you have default passwords in your database, then change these 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

- Passwords must be at least eight characters, and passwords such as welcome and oracle are not allowed.
- 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.

Related Topics

- Managing the Complexity of Passwords

- Guidelines for Securing User Accounts and Privileges

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`.

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 10g and then reset in Oracle Database 11g. The Oracle Database 11g 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 11g 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

You should 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. If you are in the CDB root, then restart the database (for example, `SHUTDOWN IMMEDIATE` followed by `STARTUP`). If you are in a PDB, connect to the root using the `SYSDBA` administrative privilege, and then enter the following statements:

```
ALTER PLUGGABLE DATABASE pdb_name CLOSE IMMEDIATE;
ALTER PLUGGABLE DATABASE pdb_name OPEN;
```

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 `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
```

   b. If you are in the CDB root, then restart the database (for example, `SHUTDOWN IMMEDIATE` followed by `STARTUP`). If you are in a PDB, connect to the root using the `SYSDBA` administrative privilege, and then enter the following statements:

```
ALTER PLUGGABLE DATABASE pdb_name CLOSE IMMEDIATE;
ALTER PLUGGABLE DATABASE pdb_name OPEN;
```

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:**
>
> *Learning Database New Features*

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

Turn off Traditional Auditing in Upgraded Oracle Databases

Traditional auditing is deprecated in Oracle Database 21c, and is desupported in Oracle Database 23c. Oracle recommends that you turn off traditional audit in your database and use only unified auditing.

Unified Auditing and Traditional Auditing (mixed mode) has been the default auditing mode from Oracle Database 12c onward. Mixed mode auditing was offered to enable you to become familiar with Unified Auditing, and to transition from Traditional Auditing. With the deprecation of Traditional Auditing in Oracle Database 21c, Oracle recommends that you turn off traditional audit in your database and use only unified auditing.. Refer to the procedure in *Oracle Database Security Guide*.

- Understanding Auditing for Oracle Database
  Decide which audit policies you want to use in the upgraded database.

- Turning Off Traditional Auditing and Using Unified Auditing for Oracle Database
  Use this procedure for multitenant container (CDB) databases to turn off traditional auditing, and to use unified auditing.
- About Managing Earlier Audit Records After You Move to Unified Auditing
  Review, archive, and purge earlier audit trails in preparation for using the unified audit trail.
- Moving From Pure Unified Auditing to Mixed-Mode Auditing
  Use this procedure to turn on traditional auditing in mixed-mode audit configuration.
- 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.

Related Topics

- Checking if Your Database Has Migrated to Unified Auditing

Understanding Auditing for Oracle Database

Decide which audit policies you want to use in the upgraded database.

For newly created databases, the mixed-mode method of unified auditing is enabled by default, which has both traditional auditing and unified auditing. The predefined audit policies `ORA_SECURECONFIG` and `ORA_LOGIN_LOGOUT` policies are enabled by default. Oracle recommends that you consider turning off traditional auditing and use only unified

If you have upgraded from an earlier release to Oracle Database 12c, then your database uses the same auditing functionality that was used in the earlier release. Oracle recommends that you consider turning off traditional auditing, and ensure the predefined audit policies `ORA_SECURECONFIG` and `ORA_LOGIN_LOGOUT` policies are enabled so that you can start using pure unified auditing.

To enable and configure the audit policies and how they are used, choose one method as follows:

- Use the pure unified audit facility.

  Transition to unified auditing to use the full unified auditing facility features. After you complete the procedure to transition 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 assists you transition to unified auditing. It 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.

Related Topics

- Auditing Activities with the Predefined Unified Audit Policies
- Secure Options Predefined Unified Audit Policy

Turning Off Traditional Auditing and Using Unified Auditing for Oracle Database

Use this procedure for multitenant container (CDB) databases to turn off traditional auditing, and to use unified auditing.

Perform the following procedure in the `root`. The procedure configures both the `root` CDB and any associated PDBs to use unified auditing.

> **Note:**
>
> Oracle recommends that you start using unified auditing now. It is deprecated in Oracle Database 21c, and desupported in Oracle Database 23ai.
>
> If you need to continue using traditional auditing as a transition, you can disable unified auditing from the container database (CDB) root only, not for individual pluggable databases (PDBs).
>
> However, when unified auditing is disabled, individual PDBs can use the mixed mode auditing, depending on whether or not the local audit policy is enabled in that PDB. If you have a CDB common audit policy enabled, then all PDBs use mixed mode auditing.

1. Log in to SQL*Plus as user `SYS` with the `SYSDBA` privilege.

   ```
   sqlplus sys as sysdba
   Enter password: password
   ```

   In the multitenant 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 Real Application Clusters (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 oracle ORACLE_HOME=$ORACLE_HOME
     ```

   - Microsoft 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 Microsoft 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
```

After you migrate to unified auditing, refer to "LOB Columns of Database Audit Trails should use Securefile Storage (Doc ID 2659172.1)" and review information about the Oracle home script *Oracle_home*`/rdbms/admin/auditpostupgrade.sql`. To obtain performance benefits of unified auditing, Oracle strongly recommends that you run this script after completing the upgrade.

Related Topics

- LOB Columns of Database Audit Trails should use Securefile Storage (Doc ID 2659172.1)

About Managing Earlier Audit Records After You Move to Unified Auditing

Review, archive, and purge earlier audit trails in preparation for using the unified audit trail.

After you complete the procedure in Oracle Database to turn off traditional auditing and 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.

Related Topics

- Archiving the Audit Trail
- Purging Audit Trail Records

Moving From Pure Unified Auditing to Mixed-Mode Auditing

Use this procedure to turn on traditional auditing in mixed-mode audit configuration.

If you decide that you want to re-enable traditional auditing in mixed-mode, then you can use this procedure to turn on traditional auditing. In this case, your database uses the mixed-mode audit facility.

> **Note:**
>
> Be aware that traditional auditing is deprecated, and is desupported in Oracle Database 23c. Plan accordingly.

1. Stop the database.

```
sqlplus sys as sysoper
Enter password: password
```

```
SQL> SHUTDOWN IMMEDIATE
SQL> EXIT
```

For Microsoft 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.
   - **Linux/Unix:** Run the following command:

     ```
     make -f ins_rdbms.mk uniaud_off oracle 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 auditing. Be aware that traditional auditing is deprecated in Oracle Database 21c and desupported in Oracle Database 23c. Plan accordingly.

Oracle Database Security 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 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.

> **Note:**
>
> You can configure AutoUpgrade to perform this task for you. To use AutoUpgrade for this task, specify the configuration file parameter `rman_catalog_connect_string`, and specify the connect string to the catalog database hosting the RMAN catalog schema. When you run AutoUpgrade, specify the RMAN username and password using the AutoUpgrade command-line parameter `load_password` with the option `group rman`. As AutoUpgrade runs, it will prompt you to provide the RMAN username and password, which AutoUpgrade then loads into its keystore, so that AutoUpgrade can connect to the RMAN catalog schema and upgrade the recovery catalog schema.

> **See Also:**
>
> - Maintaining RMAN Backups and Repository Records
> - Upgrading the Recovery Catalog
> - My Oracle Support RMAN Compatibility Matrix (Doc ID 73431.1)

Upgrading the Time Zone File Version After Upgrading Oracle Database

If the AutoUpgrade preupgrade report instructs you to upgrade the time zone files after completing the database upgrade, and you do not set AutoUpgrade to complete this task for you, then use any of the supported methods to upgrade the time zone file.

By default, AutoUpgrade changes the database time zone to the latest available level. If you don't want the time zone to be upgraded, then you must explicitly set the local parameter `timezone_upg` in your AutoUpgrade configuration file to `no`. For example:

```
upg1.timezone_upg=no
```

> **Note:**
>
> If you explicitly disable the time zone file upgrade in your AutoUpgrade configuration file, then Oracle recommends that you perform this task either as part of your upgrade plan, or at a later point in time.

Related Topics

- Datetime and Time Zone Parameters and Environment Variables
- Primary Note DST FAQ : Updated DST Transitions and New Time Zones in Oracle RDBMS and OJVM Time Zone File Patches (Doc ID 412160.1)

Enabling Disabled Release Update Bug Fixes in the Upgraded Database

Because bug fixes in Release Updates that can cause execution plan changes are disabled, Oracle recommends that you enable the disabled bug fixes that you want to use.

After you upgrade your database, the bug fix patches that can cause execution plan changes included in the Release Updates are installed disabled by default. These bug fixes will not be activated until you enable the fixes. You can either enable these fixes manually, with `PFILE` or `ALTER SYSTEM` commands, or you can use the `DBMS_OPTIM_BUNDLE` package. Starting with AutoUpgrade 19.12, the `DBMS_OPTIM_BUNDLE` package includes 58 standard fixes. You can now add additional fixes using `DBMS_OPTIM_BUNDLE`. If you add fixes, then the fixes that you add are run in addition to the default fixes.

Oracle strongly recommends that you enable these disabled patches that you want to use in your production system, and run complete workload performance tests using these patches as part of your upgrade test plan.

For more information about using `DBMS_OPTIM_BUNDLE` to enable patches that were disabled because they can change execution plans, see *Oracle Database PL/SQL Packages and Types Reference*, and My Oracle Support note 2147007.1.

Related Topics

- DBMS_OPTIM_BUNDLE
- My Oracle Support Doc ID 2147007.1 Managing "installed but disabled" bug fixes in Database Release Updates using DBMS_OPTIM_BUNDLE

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.

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.

> **Note:**
>
> 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.

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.

> **Note:**
>
> If you completed your upgrade using the AutoUpgrade utility, then you only have to complete the following:
>
> - "Identifying and Copying Oracle Text Files to a New Oracle Home"
> - "Upgrading the Oracle Clusterware Configuration"

- Changing Passwords for Oracle Supplied Accounts
  Oracle recommends that you carry out these tasks to protect new Oracle user accounts.
- 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;
```

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 Real Application Clusters (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

- Overview of Initialization Parameter Files in Oracle RAC

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.

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 = '19.1.0' 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 19.1.0`, enter the following in the initialization parameter file:

      ```
      COMPATIBLE = 19.1.0
      ```

   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`.

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.

Oracle's automated upgrade utilities perform changes to network naming and listeners during automatic upgrades. However, during a manual upgrade, neither `tnsnames.ora` nor the listeners are changed.

Related Topics

- Configuring the tnsnames.ora File After Installation

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.

8

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.

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.

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

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.

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

- *Pro*C/C++ Developer's Guide*
- *Oracle Call Interface Developer's Guide*\

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*

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:**
>
> *Learning Database New Features* 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.

9

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.

- 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.
- Prepare to Downgrade a Standby Database with the Primary
  If you are using an Oracle Data Guard Standby database, then review the procedure you can use to downgrade the standby database with the primary database
- Check COMPATIBLE Parameter when Downgrading Oracle Database
  If `COMPATIBLE` has been changed after the upgrade, then it is no longer possible to downgrade an Oracle 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.
- Using Scripts to Downgrade Oracle Database 21c
  To automate downgrades, Oracle provides the `dbdowngrade` utility script. When necessary, you can also continue to run `catdwgrd.sql` manually, as in previous releases.
- 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 APEX
  Use this procedure to avoid `INVALID OBJECTS OWNED BY APEX_050000` errors when you downgrade PDBs that contain Oracle APEX (formerly 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 21c to Oracle Database 19c, Oracle Database 18c, and Oracle Database 12c Release 2.

You can downgrade a PDB or CDB from Oracle Database 21c to Oracle Database 19c, Oracle Database 18c, or Oracle Database 12c Release 2 (12.2).

> **Note:**
>
> Starting with Oracle Database 21c, non-CDB architecture is desupported. You must upgrade a non-CDB Oracle Database to a PDB on a CDB.

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 9-1 Supported Releases and Editions for Downgrading

Oracle Database Release or Edition	Downgradable (Yes/No)	Notes
19	Yes	No additional information at this time.
18	Yes	Release Update Patch 32524155: DATABASE RELEASE UPDATE 18.14.0.0.0 (Patch) or later for your platform. Refer to "Performing Required Predowngrade Steps for Oracle Database" for details.
12.2	Yes	For PDBs, Release Update Patch DATABASE APR 2021 RELEASE UPDATE 12.2.0.1.210420 (Patch) or later for your platform. Refer to "Performing Required Predowngrade Steps for Oracle Database" for details.
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

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. Review "Primary Note for Database Proactive Patch Program (Doc ID 888.1)" on My Oracle Support for your release.

The minimum compatibility setting for Oracle Database 21c is 12.2. You cannot downgrade to releases earlier than the minimum compatibility setting for the new Oracle Database release.

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.

- 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

- My Oracle Support Doc ID 888.1
- READ and SELECT Object Privileges in *Oracle Database Security Guide*

Prepare to Downgrade a Standby Database with the Primary

If you are using an Oracle Data Guard Standby database, then review the procedure you can use to downgrade the standby database with the primary database

You can downgrade a database without breaking the standby database. To downgrade Oracle Database when a physical or logical standby database is present in the Oracle Data Guard configuration, use the procedure for your downgrade scenario as described in " Patching, Upgrading, and Downgrading Databases in an Oracle Data Guard Configuration" in *Oracle Data Guard Concepts and Administration*

Related Topics

- Patching, Upgrading, and Downgrading Databases in an Oracle Data Guard Configuration

Check COMPATIBLE Parameter when Downgrading Oracle Database

If `COMPATIBLE` has been changed after the upgrade, then it is no longer possible to downgrade an Oracle 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.

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.

If time does not allow for you to complete a full level 0 backup, then at least complete a level 1 backup.

Related Topics

- Backing Up the Database in *Oracle Database Backup and Recovery User's Guide*

Using Scripts to Downgrade Oracle Database 21c

To automate downgrades, 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 Oracle Databases To an Earlier Release
  To downgrade to a previous database release, Oracle recommends that you run the downgrade script `dbdowngrade`.
- Downgrading a CDB or Non-CDB Oracle Database Manually with catdwgrd.sql
  When you prefer to downgrade Oracle Database manually, or if you are concerned about excessive thread issues, you can run the manual `catdwgrd.sql` script.

Using Dbdowngrade to Downgrade Oracle Databases To an Earlier Release

To downgrade to a previous database release, Oracle recommends that you run the downgrade script `dbdowngrade`.

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. If you prefer to be in control of the number of resources used for a downgrade, then you can run the `catdwgrd.sql` script manually, as in previous releases. After you downgrade to the earlier database release, you can then appy any release update to that earlier release.

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 21c to Oracle Database 19c, Oracle Database 18c, or Oracle Database 12.2, 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.

> **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`.

4. 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
```

5. 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

6. 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"
   ```

7. 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.

8. Start up the database in the lower version Oracle home in Upgrade mode. 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 -e -n 1 -
     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.

9. 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.

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.

These scripts create log files. If you run `dbdowngrade` with the `-l` *`filepath`*, where *`filepath`* is the path where you want log files created, then `dbdowngrade` creates the directory you specify, and places log files there. For example:

```
./dbdowngrade -l /databases/downgrade/logs
```

If you do not specify a directory for log files, then the log files produced by the downgrade scripts are placed under the first directory found of one of these three options, in order of precedence:

- 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/21.0.0/dbhome_1
```

In this example, the $ORACLE_BASE directory is `/u01/app/oracle/product/21.0.0/dbhome_1`, and the logs are located in `/u01/app/oracle/product/21.0.0/dbhome_1/cfgtoollogs/downgrade`. In the directory, the log files are prefixed with the string `catdwgrd`.

To further manage how the `dbdowngrade` script runs, you can specify the following additional options:

- `-d` *`directory-path`* Specify the directory path, defined by *`directory-path`*, where you want the `catdwgrd.sql` file placed
- `-e` Specify that you want to turn echo off while `catdwgrd.sql` runs (the default is set to on).
- `-n` *`number`* Specify the number of parallel processes you want the `dbdowngrade` command to use. By default, the number of processes is equal to the number of CPUs divided by 2 (*`cpu_count`*`/2`).

- `-b log-file-name-base` Specify a different base file name (the value you provide for the variable *`log-file-name-base`*) for log files generated by the manual downgrade script `catdwgrd`. If you do not specify a different base file name, then the default file base name is `catdwgrd`.
- `-h` Specify that you want dbdowngrade to display a list of command options. The dbdowngade script then outputs command options to the screen, and exits.

> **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

When you prefer to downgrade Oracle Database manually, or if you are concerned about excessive thread issues, you can run the manual `catdwgrd.sql` script.

You can use the manual `catdwgrd.sql` script to downgrade Oracle Database to an earlier a supported major release, or an earlier release update.
If you are downgrading from Oracle Database 21c to Oracle Database 19c, Oracle Database 18c, or Oracle Database 12.2, 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 21c, non-CDB architecture is desupported. You must upgrade a non-CDB Oracle Database to a PDB on a CDB.

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 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:

```
SQL> SPOOL downgrade.log
```

8. Use the following command to start the downgrade, depending on your configuration:

   - Non-CDB:

```
SQL> @catdwgrd.sql
```

- CDB, using `catdwgrd` as the log file base:

```
$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 Oracle Database `OracleService`*`SID`* Oracle service, 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, using log file base `catrelod`:

```
 $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, using log file base `utlrp`:

  ```
  $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.

If you are downgrading back to Oracle Database 18c or Oracle Database 19c, and if Oracle APEX (formerly Application Express) exists in the database, then after `ultrp.sql` completes running, run the script `sys.validate_apex` to validate the APEX downgrade. You must validate APEX manually only for Oracle Database 18c or Oracle Database 19c. For other releases, running `sys.validate_apex` manually is not required.

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: 19.0.0)

    > **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*

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 19c:

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 20c 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 19c 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 19c PDB

In this step you plug the downgraded 19c PDB into the 19c 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 19c CDB.
2. Plug in the 19c 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 APEX

Use this procedure to avoid `INVALID OBJECTS OWNED BY APEX_050000` errors when you downgrade PDBs that contain Oracle APEX (formerly Application Express).

After you downgrade the PDB to an earlier release, enter a SQL statement similar to the following to drop the Oracle APEX user:, where the log file base is `drop_apex5`:

```
$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.

- 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.

- Reenabling Oracle RAC After Downgrading Oracle Database
  After the downgrade, you can re-enable Oracle Real Application Clusters (Oracle RAC).
- Reenabling 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 APEX to the Earlier Release
  After a downgrade, if you upgraded Oracle APEX (formerly Oracle Application Express) at the same time as you upgraded Oracle Database, then you must complete steps to revert to the earlier 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.
- Checking Validity of Registry Components After Downgrade
  Check the validity of registry components and identify any invalid components.

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.

Reenabling Oracle RAC After Downgrading Oracle Database

After the downgrade, you can re-enable Oracle Real Application Clusters (Oracle RAC).

For downgrades with Oracle RAC databases, where you set `CLUSTER_DATABASE=FALSE` for the downgrade, you can now set `CLUSTER_DATABASE=TRUE` again and start with all instances in the RAC cluster.

Reenabling 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 databaseTo use Oracle Database Vault after downgrading, you must register it to reenable it.

Related Topics

- Registering 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 APEX to the Earlier Release

After a downgrade, if you upgraded Oracle APEX (formerly Oracle Application Express) at the same time as you upgraded Oracle Database, then you must complete steps to revert to the earlier release.

To complete the downgrade of Oracle APEX after a database downgrade, complete all the steps listed in *Oracle APEX Installation Guide* to revert your Oracle APEX 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 APEX at the same time that you upgraded the database.

Related Topics

- *Oracle APEX 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, after a downgrade:

- 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*

Checking Validity of Registry Components After Downgrade

Check the validity of registry components and identify any invalid components.

After the downgrade, check the state of the components in the database. If you downgrade a complete CDB, then you can use the `CDB_REGISTRY` view. You can also check an individual PDB using `DBA_REGISTRY`. Ensure that all components are either `VALID` or `OPTION OFF`.

Example 9-1 Check Registry on a CDB with CDB_REGISTRY View

```
set line 200
set pages 1000
col COMP_ID format a8
col COMP_NAME format a34
col SCHEMA format a12
col STATUS format a10
col VERSION format a12
col CON_ID format 99

select CON_ID, COMP_ID, comp_name, schema, status, version from CDB_REGISTRY
order by 1,2;
```

Example 9-2 Check Registry on a Non-CDB or PDB with DBA_REGISTRY View

```
set line 200
set pages 1000
col COMP_ID format a8
col COMP_NAME format a34
col SCHEMA format a12
col STATUS format a10
col VERSION format a12
col CON_ID format 99

select CON_ID, COMP_ID, comp_name, schema, status, version from DBA_REGISTRY
order by 1,2;
```

Related Topics

- Upgrade Your Database Now! Database Migration from non-CDB to PDB – The Component Pitfall
- CHECK_COMPONENTS.SQL Script - Displays installed components from DBA_REGISTRY

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.

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 9-3 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*

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*

Errors Downgrading Databases with Oracle Messaging Gateway

If you downgrade a database configured with Oracle Messaging Gateway , then you can encounter `ORA-02303` errors.

If you downgrade an Oracle Database that contains Oracle Messaging Gateway objects, then you can encounter the following error:

```
ORA-02303: cannot drop or replace a type with type or table dependents
```

- **Cause** The `catrelod.sql` script is attempting to reload Oracle Messaging Gateway objects of a different type than the earlier Oracle Database release.
- **Action** No action. You can ignore this error.

10

Oracle Database Changes, Desupports, and Deprecations

To assist you with developing a long-term plan for your database patching and upgrades, Oracle provides this list of changes, deprecations and desupports across multiple releases.

This document is updated regularly to reflect information Oracle can provide about releases as they happen, so that you can plan for changes in releases that you plan to move to at a later date. For example, if you are upgrading to Oracle Database 19c, it can be helpful for your long-term planning to be aware of features that have been desupported in a later release, or for which there is a deprecation notice in a later release. In addition, it can be helpful to be aware that security protocols or default parameter settings have changed in a later release, so that you can begin your preparations for that change in your current release well before you want to upgrade to a release where those deprecations or desupports are in effect.

- 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
- Desupported and Deprecated Release Notice Dates
  You can identify the date that a desupport or deprecation notice has been published by using the datestamp on the topic.
- Using the ORADiff Tool to Find Release Changes
  To help you to find changes in parameter defaults, users, roles, and other changes between database releases, Oracle recommends that you use Oracle Database Release Diff Utility (ORADiff).
- Oracle Database 23ai Behavior Changes, Desupports, and Deprecations
  Review for descriptions of Oracle Database 23ai release changes.
- Oracle Database 21c Behavior Changes, Desupports, and Deprecations
  Review for descriptions of Oracle Database 21c release changes.
- Oracle Database 19c Behavior Changes, Desupports, and Deprecations
  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.
- Oracle Database 12c Release 2 (12.2) Behavior Changes, Desupports, and Deprecations
  Review for descriptions of Oracle Database 12c Release 2 (12.2) changes.
- Oracle Database 12c Release 1 (12.1) Behavior Changes, Desupports, and Deprecations
  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 the Oracle Database release for which the deprecation notice is published.
- **Desupported** features are features that are no longer supported by fixing bugs related to that feature in the Oracle Database release for which the desupport notice is published. 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.

Desupported and Deprecated Release Notice Dates

You can identify the date that a desupport or deprecation notice has been published by using the datestamp on the topic.

September 2023

The text above this sentence is a datestamp. It indicates that this topic was added with the September 2023 update to upgrade guides. Starting with August 2023, Oracle is adding this timestamp to the desupport and deprecation topics.

Using the ORADiff Tool to Find Release Changes

To help you to find changes in parameter defaults, users, roles, and other changes between database releases, Oracle recommends that you use Oracle Database Release Diff Utility (ORADiff).

August 2023

The Oracle Database Release Diff Utility (ORADiff) is an Oracle tool that allows you to compare two Oracle Database releases, or between release updates. It reports differences in parameters, privileges, users and roles, reserved words, patch fixes, database homes, objects, and more. It can also be used to obtain an inventory of these items for a particular release. ORAdiff data is refreshed when a new Release Update is released, which is every quarter. You can also generate a report that you can use to identify items that have been removed or changed from your source database release to the target database release. Check it out!

To use ORADiff, you must log in with your Oracle account Single Sign-On (SSO) user.

Related Topics

- ORADiff
- ORAdiff is live – compare two Oracle Database and Patch releases
- ORAdiff - Find the differences between two Oracle Database releases

Oracle Database 23ai Behavior Changes, Desupports, and Deprecations

Review for descriptions of Oracle Database 23ai release changes.

- Behavior Changes for Oracle Database 23ai Upgrade Planning
  Review these selected behavior changes to help plan for upgrades to Oracle Database 23ai.
- Desupported Features in Oracle Database 23ai
  As part of your upgrade plan, review the desupported features in this Oracle Database release.

- Desupported Parameters in Oracle Database 23ai
  As part of your upgrade plan, review the initialization parameters that are not supported starting with this Oracle Database release.
- Deprecated Features in Oracle Database 23ai
  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 Views in Oracle Database 23ai
  As part of your upgrade plan, review the views that are deprecated starting with this Oracle Database release.
- Deprecated Parameters in Oracle Database 23ai
  As part of your upgrade plan, review the initialization parameters listed here that are deprecated starting with this Oracle Database release.

Behavior Changes for Oracle Database 23ai Upgrade Planning

Review these selected behavior changes to help plan for upgrades to Oracle Database 23ai.

Behavior changes can include default changes for parameters or features, product name changes, and other changes to the database configuration that may require your attention.

- Oracle Database Release Number Changes
  The data provided in `VERSION`, `VERSION_LEGACY` and `VERSION_FULL` release number values are updated in Oracle Database 23ai.
- Oracle Spatial and Obsolete Objects
  If you previously used Oracle Multimedia or Oracle Locator for media objects, then Oracle Spatial is installed during the upgrade to Oracle Database 23ai if `MDSYS`-owned Locator objects exist in the database.
- REST APIs for AutoUpgrade
  To facilitate safe and secure remote use of the AutoUpgrade utility for Oracle Database upgrades, AutoUpgrade now provides REST APIs (ORDS and OCI).
- XML JSON Search Index Enhancements
  Starting with Oracle Database 23ai, enhancements to JSON search can be a factor in your upgrade planning.
- SQL/JSON Function JSON_VALUE With a Boolean JSON Value
  Starting with Oracle Database 23ai, the data type `BOOLEAN` is added to Oracle SQL. This feature extends SQL/JSON operators to allow returning a `BOOLEAN` value or accepting one as input.
- Migrate from Non-AES Algorithms in FIPS Before Upgrade
  If you use Transparent Data Encryption (TDE), then you must migrate your source database to AES encryption before starting your upgrade to Oracle Database 23ai.
- Oracle OLAP Deprecation Extended
  Analytic workspaces, the OLAP DML programming language, financial reporting, and the OLAP Java API continue to be deprecated in Oracle Database 23ai.
- About Read-Only Oracle Homes
  Starting with Oracle Database 21c, an Oracle Database installation configures all Oracle Database homes in read-only mode by default.
- SYSDATE and SYSTIMESTAMP Reflect PDB Time Zone
  Starting with Oracle Database 23ai, `SYSDATE` and `SYSTIMESTAMP` can be managed separately for each database, following the individual database time zone setting.

- Oracle Spatial GeoRaster JPEG Compression on 4-band Raster Blocks
  Starting with Oracle Database 23ai, JPEG compression on Oracle Spatial GeoRaster object can only be applied on 1-band and 3-band raster blocks.
- Document-Identifier Field Names for Duality Views Requirements
  Starting with Oracle Database 23ai, release update 23.4, when you are using duality views, the document-identifier field name must be `_id`.
- BIGFILE Is the Default for SYSAUX, SYSTEM, and USER Tablespaces
  Starting with Oracle Database 23ai, newly created databases use BIGFILE as the default for the `SYSTEM`, `SYSAUX`, and `USER` tablespaces.
- Terminal Release of Stored Outlines
  Oracle Database 23ai is the terminal release of stored outlines. Migrate all stored outlines to SQL plan baselines.

Oracle Database Release Number Changes

The data provided in `VERSION`, `VERSION_LEGACY` and `VERSION_FULL` release number values are updated in Oracle Database 23ai.

Base Version release

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 2023 has the major release version of 23, and thus its version release is 23.0.0.0.0. This base version release number is not updated over the course of the release. You can identify the base release by logging in to SQL*Plus and entering `SELECT BANNER FROM V$VERSION` to see the release displayed. This is the value associated with the `COMPATIBLE` initialization parameter.

Version Full release

The `version_full` releases are categorized by five numeric segments separated by periods, which designate the base database release, the release update, the refresh of the release update, the release month, and the release year. This value is visible in trace files. You can see the difference in the following example, comparing the same SQL command output between Oracle Database 23ai and 19c. In the Oracle Database 23ai output, the `VERSION_FULL` value provides the base release (23), the release update (4), the refresh of the release update (0), and the year and month (24.03), while the Oracle Database 19c output for `VERSION_FULL` indicates only the base release and the release update (19.8):

```
SQL> select INSTANCE_NAME,VERSION,VERSION_LEGACY,VERSION_FULL from v$instance;

INSTANCE_NAME    VERSION           VERSION_LEGACY    VERSION_FULL
---------------- ----------------- ----------------- -----------------
orcl             23.0.0.0.0        23.0.0.0.0        23.4.0.24.03


19c
-----
SQL> select INSTANCE_NAME,VERSION,VERSION_LEGACY,VERSION_FULL from v$instance;

INSTANCE_NAME    VERSION           VERSION_LEGACY    VERSION_FULL
---------------- ----------------- ----------------- -----------------
orcl19800        19.0.0.0.0        19.0.0.0.0        19.8.0.0.0
```

Related Topics

- About Oracle Database Release Numbers

Oracle Spatial and Obsolete Objects

If you previously used Oracle Multimedia or Oracle Locator for media objects, then Oracle Spatial is installed during the upgrade to Oracle Database 23ai if `MDSYS`-owned Locator objects exist in the database.

If Oracle Spatial is not needed, then to avoid triggering a Spatial installation during upgrade, remove Locator objects before you start the upgrade. To remove Oracle Spatial, run `$ORACLE_HOME/md/admin/deinssdo.sql` in the earlier release Oracle home.

If Oracle Spatial is needed, then to reduce downtime, install Oracle Spatial manually before starting the upgrade. To install Oracle Spatial, run `$ORACLE_HOME/md/admin/mdinst.sql` in the earlier release Oracle home.

- For upgrades to Oracle Database 21c, if the `MDSYS` schema existed in the database, but Oracle Spatial (SDO) did not, then Oracle Locator (LCTR) was installed into the database registry.
- During upgrades to Oracle Database 23ai, Oracle Multimedia (`ORDIM`) is removed. LCTR is also removed if it exists in the database registry. If Locator objects in the MDSYS schema exist in the database, and SDO does not, then SDO is installed.
- If you downgrade from Oracle Database 23ai, then `ORDIM` is not restored as part of the downgrade.
- If you downgrade from Oracle Database 23ai to Oracle Database 21c, then LCTR is reinstalled.
- If you are plugging in a lower version non-CDB or PDB with `ORDIM`, or `LCTR`, or both in its database registry into a target 23ai CDB where SDO is not installed, then install SDO in the Oracle Database 23ai `CDB$ROOT` before the plugin.

Related Topics

- Changes in This Release for Oracle Spatial and Graph Developer's Guide
- Pitfall: Upgrade to 21c fails when ORDIM is Present but no SDO
- My Oracle Support Doc ID 2555923.1

REST APIs for AutoUpgrade

To facilitate safe and secure remote use of the AutoUpgrade utility for Oracle Database upgrades, AutoUpgrade now provides REST APIs (ORDS and OCI).

The Oracle REST Data Services (ORDS) database API is a database management and monitoring REST API embedded into Oracle REST Data Services. The Oracle Cloud Infrastructure (OCI) REST API is enabled by configuring the REST Adapter connection to use the OCI Signature Version 1 security policy. You can now use these features to run AutoUpgrade upgrades remotely over SSH. AutoUpgrade 22 and later versions support the use of the AutoUpgrade REST APIs.

Related Topics

- https://docs.oracle.com/en/database/oracle/oracle-database/19/dbrst/

XML JSON Search Index Enhancements

Starting with Oracle Database 23ai, enhancements to JSON search can be a factor in your upgrade planning.

After you complete upgrade testing, to take advantage of the XML Search Index enhancements introduced in Oracle Database 23ai, you may want to plan to recreate your Oracle Text indexes and JSON Search Indexes after you update the `COMPATIBLE` parameter to `23.0.0.0`. However, there are no functionality changes to existing XQuery Full Text Indexes. These indexes will remain usable even after updating your `COMPATIBLE` setting.

After updating your `COMPATIBLE` parameter setting, existing JSON Search Indexes will be treated as including all available paths. Existing JSON Search Indexes that you do not recreate will remain usable. Path-aware JSON Search Indexes were introduced in Oracle Database 21c, so upgrade changes to JSON Search Indexes to enable path awareness are already handled as part of the upgrade to Oracle Database 21c. The XML Search Index and `INCLUDE` path clause features introduced with Oracle Database 23ai will require the database compatible setting to be set to `23.0.0.0.0`.

The indexing syntax you used in previous releases continues to be usable. However, syntax introduced with Oracle Database 23ai for existing indexing features will also become available, even with no compatibility setting change. However, for JSON Search Indexes, the `VALUE` mode and path-subsetting features will only be available after you create the index on a JSON type column.

After you upgrade `COMPATIBLE` to `23.0.0.0.0`, downgrade of the database is prevented in the following scenarios:

- In the presence of XML Search Indexes
- In the presence of XML or JSON Search Indexes using path-subsetting.

SQL/JSON Function JSON_VALUE With a Boolean JSON Value

Starting with Oracle Database 23ai, the data type `BOOLEAN` is added to Oracle SQL. This feature extends SQL/JSON operators to allow returning a `BOOLEAN` value or accepting one as input.

After you update the `COMPATIBLE` parameter to `23.0.0.0`, SQL/JSON path-expression item methods `boolean()` and `booleanOnly()` now return a `SQL BOOLEAN` value. This means that, in a query that has `json_value` semantics, a value produced by the item method is of type `BOOLEAN` by default: the value is handled as if it were controlled by a `json_value RETURNING` clause of type `BOOLEAN`.

In earlier releases, these methods returned a `VARCHAR2(20)` value `'true'` or `'false'`. If you need to obtain a `VARCHAR2` value (for compatibility reasons, for example), then you can wrap the value with SQL function `to_char`.

Related Topics

- *Oracle Database JSON Developer's Guide*Using SQL/JSON Function JSON_VALUE With a Boolean JSON Value

Migrate from Non-AES Algorithms in FIPS Before Upgrade

If you use Transparent Data Encryption (TDE), then you must migrate your source database to AES encryption before starting your upgrade to Oracle Database 23ai.

In Oracle Database 23ai, when you use Transparent Data Encryption (TDE) configured for the Federal Information Processing Standard (FIPS), only Advanced Encryption Standard (AES) ciphers AES-128, AES-192, and AES-256 are allowed. If your source Oracle Database is configured for the FIPS mode, and it is using any other algorithms to encrypt a column or a tablespace, then the column and tablespace must be rekeyed using AES before upgrade.

If you upgrade your source database and it is using desupported algorithms for encryption, then the upgraded database will either fail to start up, or encrypted tablespaces will not be available, because the database cannot decrypt tablespace keys. In that case, Oracle recommends that you downgrade your database, upgrade your encryption keys to the supported AES ciphers, and then restart the upgrade.

Oracle OLAP Deprecation Extended

Analytic workspaces, the OLAP DML programming language, financial reporting, and the OLAP Java API continue to be deprecated in Oracle Database 23ai.

Date: July 2024

Be aware that OLAP will not be supported beyond the term of the current release (Oracle Database 23ai) premier support. Oracle strongly recommends that you do not start new projects using OLAP and begin migrating applications using OLAP to alternatives now. If your application requires an in-database dimensional model, then consider using Oracle Analytic Views. Analytic views provide a dimensional semantic model, calculations, and query semantics using data in Oracle Database. When used with columnar tables, analytic views provide query performance similar to the OLAP Option. If your application requires support for advanced dimensional analytics, what-if analysis, or forecasting, then consider Oracle Essbase. Oracle Essbase is a multidimensional database management system with support for complex dimensional business analytics.

About Read-Only Oracle Homes

Starting with Oracle Database 21c, an Oracle Database installation configures all Oracle Database homes in read-only mode by default.

A read-only Oracle Home simplifies provisioning by implementing separation of installation and configuration.

Before Oracle Database 21c, the default `ORACLE_HOME` layout combined `ORACLE_HOME`, `ORACLE_BASE_HOME` and `ORACLE_BASE_CONFIG` into a single location. Starting with Oracle Database 21c, the only available configuration is a read-only `ORACLE_HOME` where `ORACLE_BASE_HOME` and `ORACLE_BASE_CONFIG` are located separately from `ORACLE_HOME`.

In a read-only Oracle home, all the configuration data and log files reside outside of the read-only Oracle home.

Apart from the traditional `ORACLE_BASE` and `ORACLE_HOME` directories, the following directories contain files that used to be in `ORACLE_HOME`:

- `ORACLE_BASE_HOME`
- `ORACLE_BASE_CONFIG`

> **Note:**
>
> This feature does not affect how database administrators monitor, diagnose, and tune their system performance.

SYSDATE and SYSTIMESTAMP Reflect PDB Time Zone

Starting with Oracle Database 23ai, `SYSDATE` and `SYSTIMESTAMP` can be managed separately for each database, following the individual database time zone setting.

To increase the applicability and transparency of multitenant for even more consolidations of independent databases, pluggable databases (PDBs) can now be managed individually for databases in container databases (CDBs). All user-visible operations and internal functions (for example, Oracle Scheduler or Oracle Flashback technology) adhere to this setting.

In previous releases, the `SYSDATE` and `SYSTIMESTAMP` settings were centrally managed, following the operating system level of the database host. With this update, Oracle Multitenant enables an Oracle Database to consolidate multiple pluggable databases as self-contained databases, improving resource utilization and database management. If you do not change your initialization parameters after the upgrade, then the default remains, and all pluggable databases inherit the system time from the operating system.

Related Topics

- TIME_AT_DBTIMEZONE

Oracle Spatial GeoRaster JPEG Compression on 4-band Raster Blocks

Starting with Oracle Database 23ai, JPEG compression on Oracle Spatial GeoRaster object can only be applied on 1-band and 3-band raster blocks.

If you have any GeoRaster objects that have 4-band raster blocks, then you must reblock the objects to 1-band or 3-band raster blocks before applying the JPEG compression. If you have any GeoRaster objects that have JPEG compression on 4-band raster blocks in previous releases, then you must reblock them to 1-band or 3-band raster blocks before upgrading to Oracle Database 23ai.

Related Topics

- My Oracle Support Doc ID 2960620.1

Document-Identifier Field Names for Duality Views Requirements

Starting with Oracle Database 23ai, release update 23.4, when you are using duality views, the document-identifier field name must be `_id`.

The name `_id` is the required (and only) document identifier that you can use for duality views. Duality views that do not use the `_id` identifier field will fail. This restriction applies with Oracle Database 23ai (23.4) and later updates.

Related Topics

- "Document-Identifier Fields for Duality Views" in *JSON-Relational Duality Developer's Guide*

BIGFILE Is the Default for SYSAUX, SYSTEM, and USER Tablespaces

Starting with Oracle Database 23ai, newly created databases use BIGFILE as the default for the `SYSTEM`, `SYSAUX`, and `USER` tablespaces.

A **bigfile** tablespace is a tablespace with a single, but large datafile. Traditional small file tablespaces (`smallfile`), in contrast, typically contain multiple datafiles, but the files cannot be as large. Making `SYSAUX`, `SYSTEM` and `USER` tablespaces bigfile tablespaces by default will benefit large databases by reducing the number of datafiles, thereby simplifying datafile, tablespace and overall global database management for users.

Terminal Release of Stored Outlines

Oracle Database 23ai is the terminal release of stored outlines. Migrate all stored outlines to SQL plan baselines.

Stored outlines were deprecated in Oracle Database 11g. Oracle is preparing to discontinue further support for stored outlines. Alternative functionality is provided using SQL Plan Management baselines, which provide numerous enhancements.

Desupported Features in Oracle Database 23ai

As part of your upgrade plan, review the desupported features in this Oracle Database release.

- ODP.NET OracleConfiguration.DirectoryType Property and .NET Configuration File DIRECTORY_TYPE Setting Desupported
- Original Export Utility (EXP) Desupported
  The original Oracle Database Export (`exp`) utility is desupported in Oracle Database 23ai.
- MySQL Client Library Driver for Oracle Desupported
- ACFSUTIL REPL REVERSE Desupported
- Cluster Domain - Domain Services Cluster Desupported
- DBSNMP Packages for Adaptive Thresholds Feature Desupported
  DBSNMP PL/SQL packages associated with the Adaptive Thresholds feature are desupported in Oracle Database 23ai.
- Policy-Managed Database Deployment Desupported
  The policy-managed database deployment option is desupported in Oracle Database 23ai.
- Enterprise User Security User Migration Utility Desupport
  Starting with Database 23ai, the User Migration Utility (UMU) part of Enterprise User Security (EUS) is desupported.
- Oracle Enterprise Manager Database Express Desupported
  Oracle Enterprise Manager Database Express (EM Express) is desupported in Oracle Database Release 23ai.
- Oracle Wallet Manager (OWM) Desupported
  Starting with Oracle Database 23ai, the Oracle Wallet Manager (OWM) is desupported.
- RASADM Desupported
  The Real Application Security GUI administration tool (RASADM) is desupported with Oracle Database 23ai.

- Oracle Label Security Parameters and Functions Desupported
  Oracle Label Security (OLS) parameters and functions that were deprecated in Oracle Database 12c are desupported in Oracle Database 23ai.
- Oracle Internet Directory with Oracle Label Security Desupported
  The integration of Oracle Internet Directory (OID) with Oracle Label Security (OLS) is desupported with Oracle Database 23ai.
- Granting Administrative Privilege to RADIUS Users Desupported
  Starting with Oracle Database 23ai, users authenticating to the database using the legacy RADIUS API no longer are granted administrative privileges.
- Transparent Data Encryption PKI Keys Desupported
  Transparent Data Encryption (TDE) public key infrastructure (PKI) keys are desupported with Oracle Database 23ai
- GOST and SEED TDE Cryptographic Encryption Algorithms Desupported
  Starting with Oracle Database 23ai, the Transparent Data Encryption (TDE) encryption libraries for the GOST and SEED algorithms are desupported and removed. The GOST and SEED decryption libraries are deprecated. Both are removed on HP Itanium platforms.
- Oracle Database 10G Password Verifier Desupported
  Starting with Oracle Database 23ai, the 10G database password verifier is desupported.
- Transport Layer Security versions 1.0 and 1.1 Desupported
  Starting with Database 23ai, the use of Transport Layer Security protocol versions 1.0 and 1.1 are desupported.
- Unix Crypt (MD5crypt) Password Verifier Desupported
  The Unix Crypt (`MD5crypt`) password verifier algorithm is desupported in Oracle Database 23ai server and clients.
- FIPS Strength 80 Encryption Desupported
  FIPS encryption strength 80 is desupported with the release of Oracle Database 23ai. Use FIPS encryption strength 112 instead.
- Diffie-Hellman Anonymous Ciphers Desupported
  The use of Diffie-Hellman anonymous ciphers (DH anon) is desupported with Oracle Database 23ai for both outbound connections and for database client/server connections.
- Oracle Database Extensions for .NET Desupported
  Oracle Database Extensions for .NET is desupported. Oracle recommends that you either place .NET code in the middle tier, or use the External Procedures feature, or rewrite the code using PL/SQL or Java.
- Quality of Service Management Desupported
  Starting with Oracle Database Release 23ai, Oracle Quality of Service Management (QoSM, or QoS Management) is desupported.
- Traditional Auditing Desupported
  Traditional auditing is desupported in Oracle Database 23ai. Oracle recommends that you use unified auditing.
- Desupport of config.sh
  Starting with Oracle Database 23ai, the Oracle Grid Infrastructure Configuration Wizard tool (`config.sh`) is desupported. Use `gridSetup.sh` instead.
- OLS Table LABELS Column Desupport
  The `LABELS` column in the `ALL_SA_USER_LABELS` and `DBA_SA_USER_LABELS` OLS tables is desupported in Oracle Database 23ai.
- Desupport of 32-Bit Oracle Database Clients
  32-bit Oracle Database clients are desupported in Oracle Database 23ai.

- Desupport of Oracle GoldenGate Replication for Oracle Globally Distributed Database High Availability
  The use of Oracle GoldenGate Replication for Shard-Level High Availability in Oracle Globally Distributed Database (formerly known as Oracle Sharding) is desupported in Oracle Database 23ai.
- Grid Infrastructure Management Repository (GIMR) Desupported
  Starting with Oracle Database 23ai, the use of Grid Infrastructure Management Repository (GIMR) is desupported.
- Data Recovery Advisor (DRA) Desupport
  Starting in Oracle Database 23ai, the Data Recovery Advisor (DRA) feature is desupported.
- DBUA and Manual Upgrade Methods Desupported
  Database Upgrade Assistant (DBUA) is desupported. Oracle recommends using AutoUpgrade to upgrade your database.
- Desupport of Oracle Data Masking and Subsetting with Oracle Real Application Testing
  The integration of data masking with Oracle Real Application Testing is desupported in Oracle Database 23ai.
- Desupport of Shared Grid Naming Service Option for Addresses
  The Shared GNS option of Grid Naming Service for name resolution in a cluster is desupported in Oracle Grid Infrastructure 23ai.
- DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL Procedure Desupported
  The `DBMS_AUDIT_MGMT` procedure `DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT` feature is desupported in Oracle Database 23ai.
- AUDIT_TRAIL_WRITE Mode of the AUDIT_TRAIL_PROPERTY Parameter Desupported
  The DBMS_AUDIT_MGMT AUDIT_TRAIL_WRITE Mode of the AUDIT_TRAIL_PROPERTY feature is desupported in Oracle Database 23ai.
- Cluster Time Synchronization Service Desupported
  Cluster Time Synchronization Service (CTSS) is desupported in Oracle Database 23ai.
- Oracle Connection Manager Parameter (CMAN) Password Access Desupported
  The use of password access to Oracle Connection Manager parameters is desupported in Oracle Database 23ai.
- Desupport of EUS Current User Database Links
  Current user database links for Enterprise User Security (EUS) are not supported in Oracle Database 23ai
- Desupport of oracle.jdbc.rowset Package
  The Java `oracle.jdbc.rowset` package is desupported in Oracle Database 23ai.
- Desupport of ACFS on IBM AIX
  Oracle Advanced Cluster File System (ACFS) on IBM AIX is desupported in Oracle Database 23ai
- RECOVER…SNAPSHOT TIME Desupported
  The `RECOVER…SNAPSHOT TIME` method of recovering a database to a point in time using a particular snapshot is desupported in Oracle Database 23ai.

ODP.NET OracleConfiguration.DirectoryType Property and .NET Configuration File DIRECTORY_TYPE Setting Desupported

The Oracle Data Provider for .NET `DirectoryType` property that is part of the `OracleConfiguration` class is desupported in Oracle Database 23ai.

Date: April 2023

The .NET configuration file `DIRECTORY_SERVER_TYPE` setting replaces the `DIRECTORY_TYPE` setting. The `OracleConfiguration DirectoryServerType` property replaces the `DirectoryType` property. All of these properties have identical functionality. Oracle recommends to developers to use and migrate to the `DirectoryServerType` property. The `DirectoryServerType` name better aligns with the `ldap.ora` parameter, `DIRECTORY_SERVER_TYPE`, which provides equivalent functionality.

Original Export Utility (EXP) Desupported

The original Oracle Database Export (`exp`) utility is desupported in Oracle Database 23ai.

Date: April 2023

Oracle recommends that you use Oracle Data Pump Export (`expdp`).

MySQL Client Library Driver for Oracle Desupported

The MySQL Client Library Driver for Oracle is desupported in Oracle Database 23ai.
Date: April 2023

The MySQL Client library driver, `liboramysql`, was deprecated in Oracle Database 21c. It is now desupported. There is no replacement. This desupport does not affect the ability of older Oracle Database Client releases that use `liboramysql` to connect to the database. However, the features available to use through these clients eventually can be limited.

ACFSUTIL REPL REVERSE Desupported

The `acfsutil repl reverse` command is desupported in Oracle Database 23ai. Use `repl failover` or `repl switchover` instead.
Date: April 2023

The Oracle Automatic Cluster File System (ACFS) command utility `acfsutil` includes the commands `repl failover` and `repl switchover`. These commands provide more functionality, including all the functions of `acfsutil repl reverse`. For this reason, Oracle is desupporting the `acfsutil repl reverse` command.

Cluster Domain - Domain Services Cluster Desupported

Starting with Oracle Grid Infrastructure 23ai, Domain Services Clusters (DSC), which is part of the Oracle Cluster Domain architecture, are desupported.
Date: April 2023

Oracle Cluster Domains consist of a Domain Services Cluster (DSC) and Member Clusters. Member Clusters were deprecated in Oracle Grid Infrastructure 19c. The DSC continues to be available to provide services to production clusters. However, with most of those services no longer requiring the DSC for hosting, installation of DSCs are desupported in Oracle Database 23ai. Oracle recommends that you use any cluster or system of your choice for services previously hosted on the DSC, if applicable. Oracle will continue to support the DSC for hosting shared services, until each service can be used on alternative systems.

DBSNMP Packages for Adaptive Thresholds Feature Desupported

DBSNMP PL/SQL packages associated with the Adaptive Thresholds feature are desupported in Oracle Database 23ai.

Date: April 2023

Beginning with Oracle Enterprise Manager Cloud Control 13.5, all features of Database Server Adaptive Thresholds and Baseline Metric Thresholds are removed for all Oracle Database targets. The following database server side packages that support this feature are longer available: DBSNMP.BSLN, DBSNMP.BSLN_INTERNAL. DBSNMP.MGMT_RESPONSE continues to be supported.

For more information about this deprecation and desupport, refer to My Oracle Support 2697846.1

Policy-Managed Database Deployment Desupported

The policy-managed database deployment option is desupported in Oracle Database 23ai.

Date: April 2023

Policy-managed databases were deprecated in Oracle Database 21c, and admin-managed database deployment was enhanced with functions similar to policy managed databases. By converging the automation provided by policy-managed database with the consistency of an admin-managed database, Oracle seeks to simplify database management tasks for database administrators. This converged database deployment provides the best of both options, such as providing the options to rank and define the order of database startup, without requiring you to choose a specific style during deployment.

Enterprise User Security User Migration Utility Desupport

Starting with Database 23ai, the User Migration Utility (UMU) part of Enterprise User Security (EUS) is desupported.

Date: April 2023

There is no workaround.

Oracle Enterprise Manager Database Express Desupported

Oracle Enterprise Manager Database Express (EM Express) is desupported in Oracle Database Release 23ai.

Date: April 2023

EM Express is a web-based database management tool that is built inside Oracle Database. It supports key performance management and basic database administration functions. EM Express was deprecated in Oracle Database 21c. Many of EM Express's capabilities are now available in Oracle Cloud Infrastructure (OCI) Database Management service, Oracle Enterprise Manager Cloud Control, or Oracle SQL Developer.

Instead of EM Express, Oracle recommends that you choose a tool that fits the requirements and deployment type (cloud, on-premises, or hybrid) from OCI Database Management service, Oracle Enterprise Manager Cloud Control or Oracle SQL Developer Web or Oracle SQL Developer desktop products.

Oracle Wallet Manager (OWM) Desupported

Starting with Oracle Database 23ai, the Oracle Wallet Manager (OWM) is desupported.

Date: April 2023

Oracle recommends using the `orapki` command line tool to replace OWM.

RASADM Desupported

The Real Application Security GUI administration tool (RASADM) is desupported with Oracle Database 23ai.

Date: April 2023

RASADM is no longer supported for use with Oracle Database 23ai. Oracle recommends that you use the RAS PL/SQL API which includes all the functionality of RASADM.

Oracle Label Security Parameters and Functions Desupported

Oracle Label Security (OLS) parameters and functions that were deprecated in Oracle Database 12c are desupported in Oracle Database 23ai.

Date: April 2023

These previously deprecated OLS parameters and functions have alternate parameters that can be used.

The following OLS functions and parameters are desupported:

`Least_UBOUND`. Use `OLS_GREATEST_LBOUND` instead.
`LUBD`. Use `OLS_GLBD` instead.
`DOMINATES`. Use the `OLS_DOMINATES` standalone function instead.
`DOM`. Use the `OLS_STRICTLY_DOMINATES` standalone function instead.
`STRICTLY_DOMINATES`. Use the `OLS_STRICTLY_DOMINATES` standalone function instead.
`S_DOM`. Use the `OLS_STRICTLY_DOMINATES` standalone function instead.
`DOMINATED_BY`. Use the `OLS_DOMINATED_BY` standalone function instead.
`DOM_BY`. Use the `OLS_DOMINATED_BY` standalone function instead.
`STRICTLY_DOMINATED_BY`. Use the OLS_STRICTLY_DOMINATED_BY standalone function instead.
`S_DOM_BY`. Use the `OLS_STRICTLY_DOMINATED_BY` standalone function instead.

Also, the OLS standalone function `SA_UTL.DOMINATES` boolean datatype is desupported. The `SA_UTL.DOMINATES` function that uses the `NUMBER` datatype is not deprecated.

For reference about alternatives see: Oracle Label Security Administrator's Guide Deprecated Features

Oracle Internet Directory with Oracle Label Security Desupported

The integration of Oracle Internet Directory (OID) with Oracle Label Security (OLS) is desupported with Oracle Database 23ai.

Date: April 2023

The use of OID to store OLS policies and labels is desupported with Oracle Database 23ai. No replacement for this feature is planned. If you are using this feature, then you must create a custom method to copy and store this information.

Granting Administrative Privilege to RADIUS Users Desupported

Starting with Oracle Database 23ai, users authenticating to the database using the legacy RADIUS API no longer are granted administrative privileges.

Date: April 2023

In previous releases, users authenticating with RADIUS API could be granted administrative privileges such as `SYSDBA` or `SYSBACKUP`. In Oracle Database 23ai, Oracle introduces a new RADIUS API that uses the latest standards. To grant administrative privileges to users, ensure the database connection to the database uses the new RADIUS API, and that you are using the Oracle Database 23ai client to connect to the Oracle Database 23ai server.

Transparent Data Encryption PKI Keys Desupported

Transparent Data Encryption (TDE) public key infrastructure (PKI) keys are desupported with Oracle Database 23ai

Date: April 2023

The deprecation of TDE PKI keys was announced with Oracle Database 12c. Oracle recommends using one of the current TDE keys instead. Follow the documentation to rekey your database with a new TDE key before upgrading the Oracle Database 23ai. If the upgrade script detects PKI keys used with TDE, then the script will stop and notify you to rekey the TDE Master Key before upgrade.

Related Topics

- Setting or Rekeying the TDE Master Encryption Key in the Keystore

GOST and SEED TDE Cryptographic Encryption Algorithms Desupported

Starting with Oracle Database 23ai, the Transparent Data Encryption (TDE) encryption libraries for the GOST and SEED algorithms are desupported and removed. The GOST and SEED decryption libraries are deprecated. Both are removed on HP Itanium platforms.

Date: April 2023

GOST 28147-89 has been deprecated by the Russian government, and SEED has been deprecated by the South Korean government. If you need South Korean government-approved TDE cryptography, then use ARIA instead. If you are using GOST 28147-89, then you must decrypt and encrypt with another supported TDE algorithm. The decryption algorithms for GOST 28147-89 and SEED are included with Oracle Database 23ai, but are deprecated, and the GOST encryption algorithm is desupported with Oracle Database 23ai. If you are using GOST or SEED for TDE encryption, then Oracle recommends that you perform an online rekey operation before upgrading to Oracle Database 23ai. However, with the exception of the HP Itanium platform, the GOST and SEED decryption libraries are available with Oracle Database 23ai, so you can also decrypt after upgrading.

Oracle Database 10G Password Verifier Desupported

Starting with Oracle Database 23ai, the 10G database password verifier is desupported.

Date: April 2023

The database password verifier for Oracle Database 10g, `10G` is no longer supported or available on Oracle Database 23ai. Refer to the database upgrade guide preinstallation chapters for information about how to identify the Oracle Database 10G database password verifiers, and how to update the database user to use the latest and most secure database password verifier cryptography.

Parameters associated with the 10G verifiers were desupported with Oracle Database 21c:

- `IGNORECASE`
- `SEC_CASE_SENSITIVE_LOGON`

The 11G and 12C verifiers were the default password verifiers starting with Oracle Database 12c Release 2 (12.2). The following related parameter values are also desupported:

- `SQLNET.ALLOWED_LOGON_VERSION_SERVER` values of 8, 9, 10 and 11 are now removed (12 and 12a remain).
- `SQLNET.ALLOWED_LOGON_VERSION_CLIENT` value of 11 is now removed (12 and 12a remain).

In addition, note the following:

- Patched 10g and 11g clients using the 12c verifier will continue to work.
- Client capabilities listed in the Database Net Services Reference are modified.
- `O3L`, `O4L`, and `O5L` client capabilities are removed from the client capabilities table.

Transport Layer Security versions 1.0 and 1.1 Desupported

Starting with Database 23ai, the use of Transport Layer Security protocol versions 1.0 and 1.1 are desupported.

Date: April 2023

In most cases, this change will not have any impact, because the database client and server will negotiate the use of the most secure protocol and cipher algorithm. However, if TLS 1.0 or 1.1 has been specified, then you must either remove it to allow the database server and client to pick the most secure protocol, or you must specify either TLS 1.2, or TLS 1.3, or both, for the protocol. Oracle recommends using the latest, most secure protocol. That protocol is TLS 1.3, which is introduced with Oracle Database 23ai.

Unix Crypt (MD5crypt) Password Verifier Desupported

The Unix Crypt (`MD5crypt`) password verifier algorithm is desupported in Oracle Database 23ai server and clients.

Date: April 2023

Enterprise User Security (EUS) customers with users in Oracle Internet Directory (OID) potentially can be using older, less secure password verifiers generated by Unix Crypt, either by OID, or by the operating system, before they were migrated to OID. Compared to current methods to hash the password, Unix Crypt is a less secure algorithm. Oracle Database can no longer authenticate EUS or OID users with the older password verifiers. Oracle recommends that you reset passwords in OID now, using newer, more secure hashing algorithms.

FIPS Strength 80 Encryption Desupported

FIPS encryption strength 80 is desupported with the release of Oracle Database 23ai. Use FIPS encryption strength 112 instead.

Date: April 2023

FIPS encryption strength 80 is equivalent to the 1024 key lengths for RSA, Diffie-Hellman (DH), and Digital Signature Algorithm (DSA). The lower strength 0 (RSA/DH/DSA 512 key length) was desupported for FIPS use with Oracle Database 21c. The only remaining FIPS encryption strength supported for Oracle Database 23ai is 112 (RSA/DH/DSA 2048 key length). The encryption strength of 80 and 0 are still available for non-FIPS use. Oracle recommends that you use FIPS 112, because FIPS encryption strength 112 is much stronger than FIPS encryption strength 80, and the longer key lengths are used with Oracle Database 23ai.

Diffie-Hellman Anonymous Ciphers Desupported

The use of Diffie-Hellman anonymous ciphers (DH anon) is desupported with Oracle Database 23ai for both outbound connections and for database client/server connections.

Date: April 2023

Removing the DH anon ciphers improves the security for Oracle Database connections.

The following 3 ciphers are desupported with Oracle Database 23ai:

- `TLS_DH_ANON_WITH_AES_256_GCM_SHA384`
- `TLS_DH_ANON_WITH_AES_128_GCM_SHA256`
- `SSL_DH_ANON_WITH_3DES_EDE_CBC_SHA`

If you are an Enterprise User Security (EUS) customer, then you must confirm that you are using server-based TLS authentication in your OID connections. For database client/server connections, you should allow the server and client to negotiate for the strongest possible connection. If a DH anon cipher was specified, then you must remove that cipher, and the database server must be authenticated for a 1-way TLS connection.

Oracle Database Extensions for .NET Desupported

Oracle Database Extensions for .NET is desupported. Oracle recommends that you either place .NET code in the middle tier, or use the External Procedures feature, or rewrite the code using PL/SQL or Java.

Date: April 2023

Oracle Database Extensions for .NET is a feature of Oracle Database on Microsoft Windows that enables you to use stored procedures and functions written in a language managed by .NET, such as C#.

Oracle Database hosts the Microsoft Common Language Runtime (CLR) in an external process, outside of the Oracle Database process. Application developers can write stored procedures and functions using any .NET compliant language, such as C# and VB.NET, and use these .NET stored procedures in the database, in the same manner as other PL/SQL or Java stored procedures. .NET stored procedures can be called from PL/SQL packages, procedures, functions, and triggers; from SQL statements; or from anywhere a PL/SQL procedure or function can be called.

Migration options include:

- Moving the .NET code (assemblies) into a middle tier
- Using the External Procedures feature to have the external process load and execute the .NET assembly
- Rewriting the stored procedures using PL/SQL or Java

Quality of Service Management Desupported

Starting with Oracle Database Release 23ai, Oracle Quality of Service Management (QoSM, or QoS Management) is desupported.

Date: April 2023

Oracle QoSM automates the workload management for an entire system by adjusting the system configuration based on predefined policies to keep applications running at the performance levels needed. Applications and databases are increasingly deployed in systems that provide some of the resource management capabilities of QoSM. At the same time, Oracle's Autonomous Health Framework has been enhanced to adjust and provide recommendations to mitigate events and conditions that impact the health and operational capability of a system and its associated components. For those reasons, QoSM and its Memory Guard feature are desupported with Oracle Database 23ai.

Traditional Auditing Desupported

Traditional auditing is desupported in Oracle Database 23ai. Oracle recommends that you use unified auditing.

Date: April 2023

Starting with Oracle Database 23ai, unified auditing is the way forward to perform Oracle Database auditing. Unified auditing offers more flexibility to perform selective and effective auditing, which helps you focus on activities that really matter to your enterprise. Unified auditing has one single and secure unified trail, conditional policy for audit selectivity, and default preconfigured policies for simplicity. To improve security and compliance, Oracle strongly recommends that you use unified auditing.

Related Topics

- Handling the Desupport of Traditional Auditing

Desupport of config.sh

Starting with Oracle Database 23ai, the Oracle Grid Infrastructure Configuration Wizard tool (`config.sh`) is desupported. Use `gridSetup.sh` instead.

Date: April 2023

With the introduction of image-based Oracle Grid Infrastructure installation, the `config.sh` functionality has become obsolete. The `gridSetup.sh` script supports the response file functionality previously used with config.sh.

OLS Table LABELS Column Desupport

The `LABELS` column in the `ALL_SA_USER_LABELS` and `DBA_SA_USER_LABELS` OLS tables is desupported in Oracle Database 23ai.

Date: April 2023

The `LABELS` column was previously deprecated for Oracle Label Security (OLS), because it duplicates information available in the other columns. This change improves the usability of these tables, because you no longer need to choose which column you should reference.

Desupport of 32-Bit Oracle Database Clients

32-bit Oracle Database clients are desupported in Oracle Database 23ai.

Date: April 2023

Oracle has discontinued developing 32-bit Oracle Database clients. Oracle recommends that you use 64-bit Oracle Database clients. If you are using 32-bit applications, then you can

continue to use older 32-bit Oracle Database clients, subject to “Client / Server Interoperability Support Matrix for Different Oracle Versions (Doc ID 207303.1)," and the release support lifecycle (My Oracle Support 742060.1).

Related Topics

- Client / Server Interoperability Support Matrix for Different Oracle Versions (Doc ID 207303.1)
- Release Schedule of Current Database Releases (Doc ID 742060.1)

Desupport of Oracle GoldenGate Replication for Oracle Globally Distributed Database High Availability

The use of Oracle GoldenGate Replication for Shard-Level High Availability in Oracle Globally Distributed Database (formerly known as Oracle Sharding) is desupported in Oracle Database 23ai.

Date: April 2023

Grid Infrastructure Management Repository (GIMR) Desupported

Starting with Oracle Database 23ai, the use of Grid Infrastructure Management Repository (GIMR) is desupported.

Date: July 2023

Oracle is using another repository to store metadata. The use of GIMR is desupported in Oracle Database 23ai. This transition is planned so that there will be no effect on features using GIMR (GIMR clients), such as Oracle Autonomous Health Framework (AHF), Cluster Health Monitor (CHM), Oracle Cluster Health Advisor (CHA), or Oracle Fleet Patching and Provisioning (FPP).

Data Recovery Advisor (DRA) Desupport

Starting in Oracle Database 23ai, the Data Recovery Advisor (DRA) feature is desupported.

Date: July 2023

The desupport of DRA includes desupporting the following Oracle Recovery Manager (RMAN) commands: `LIST FAILURE`, `ADVISE FAILURE`, `REPAIR FAILURE`, and `CHANGE FAILURE`. Database administrators will no longer have access to these commands. There is no replacement feature for DRA.

DBUA and Manual Upgrade Methods Desupported

Database Upgrade Assistant (DBUA) is desupported. Oracle recommends using AutoUpgrade to upgrade your database.

Date: July 2023

Database Upgrade Assistant (DBUA) is desupported for any upgrades and migrations to Oracle Database 23ai, regardless of whether they are done on-premises, in any cloud, or in a hybrid approach. Oracle AutoUpgrade is the only supported tool to upgrade Oracle Database, with or without a migration to the multitenant (CDB) architecture.

Related Topics

- AutoUpgrade Tool (Doc ID 2485457.1)

Desupport of Oracle Data Masking and Subsetting with Oracle Real Application Testing

The integration of data masking with Oracle Real Application Testing is desupported in Oracle Database 23ai.

Date: November 2023

For testing, Oracle recommends that you keep the test database in the production realm for security, but create user accounts for testing that cannot access your data to run both SQL Performance Analyzer (SPA) trials and Database Replay.

Desupport of Shared Grid Naming Service Option for Addresses

The Shared GNS option of Grid Naming Service for name resolution in a cluster is desupported in Oracle Grid Infrastructure 23ai.

Date: December 2023

The Shared GNS name resolution option is a daemon running on one cluster that is configured to provide name resolution for all clusters in domains that are delegated to GNS for resolution, which can be centrally managed using SRVCTL commands. Starting with Oracle Database 23ai, both the option of using role to specify a primary or secondary role for GNS and the option of using Shared GNS are discontinued as deployment options. Existing configurations using these features will still work after upgrade, but they will not be available for new GNS deployments. Oracle recommends that you use your own DNS to configure client connections to Oracle Grid Infrastructure.

DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL Procedure Desupported

The `DBMS_AUDIT_MGMT` procedure `DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT` feature is desupported in Oracle Database 23ai.

Date: December 2023

This procedure is no longer necessary because audit records now bypass the common logging infrastructure queues and are directly written to a new internal relational table.

AUDIT_TRAIL_WRITE Mode of the AUDIT_TRAIL_PROPERTY Parameter Desupported

The DBMS_AUDIT_MGMT AUDIT_TRAIL_WRITE Mode of the AUDIT_TRAIL_PROPERTY feature is desupported in Oracle Database 23ai.

Date: December 2023

This procedure is no longer necessary because audit records now bypass the common logging infrastructure queues and are directly written to a new internal relational table.

Cluster Time Synchronization Service Desupported

Cluster Time Synchronization Service (CTSS) is desupported in Oracle Database 23ai.

Date: January 2024

To synchronize time between cluster member nodes, use either an operating system configured network time protocol such as `ntp` or `chrony`, or Microsoft Windows Time service. To verify that you have network time synchronization configured, you can use the `cluvfy comp clocksync -n allnodes` command.

Oracle Connection Manager Parameter (CMAN) Password Access Desupported

The use of password access to Oracle Connection Manager parameters is desupported in Oracle Database 23ai.

Date: March 2024

Oracle provides an enhanced connection method, Local Operating System Authentication" (LOSA), which permits only the user who started CMAN to perform admin operations. This method is consistent with other operating system authentication methods used with Oracle Database. If you are currently using password access to CMAN, then Oracle recommends that you remove the CMAN password, and instead rely on LOSA.

Desupport of EUS Current User Database Links

Current user database links for Enterprise User Security (EUS) are not supported in Oracle Database 23ai

Date: April 2024

Because Oracle Database releases after Oracle Database 19c only use the multitenant architecture, the use of EUS current user database links (`CREATE DATABASE LINK…TO CURRENT_USER`) is not supported in releases after Oracle Database 19c. Oracle recommends that you use fixed user or connected user database links with EUS.

Desupport of oracle.jdbc.rowset Package

The Java `oracle.jdbc.rowset` package is desupported in Oracle Database 23ai.

Date: May 2024

Oracle recommends that you use the Standard JDBC Rowset package to replace this feature.

Desupport of ACFS on IBM AIX

Oracle Advanced Cluster File System (ACFS) on IBM AIX is desupported in Oracle Database 23ai

Date: December 2024

To align the support for the IBM AIX platform, Oracle Advanced Cluster File System (ACFS) on IBM AIX is desupported in Oracle Database 23ai. For data or files currently stored on an ACFS file system that cannot be stored in Oracle Database, you can use the IBM General Parallel File System (GPFS).

RECOVER...SNAPSHOT TIME Desupported

The `RECOVER…SNAPSHOT TIME` method of recovering a database to a point in time using a particular snapshot is desupported in Oracle Database 23ai.

Date: February 2025

Instead of `RECOVER…SNAPSHOT TIME`, Oracle recommends that you use `ALTER DATABASE BEGIN/END BACKUP` before and after creating the storage snapshot of the data files and then use `RECOVER .. UNTIL TIME` to a specific timestamp or system change number (SCN) after the `END BACKUP` completion time. Oracle recommends that `ALTER DATABASE BEGIN/END BACKUP` always be used when performing snapshots on a running database to ensure data recovery integrity. Archived log redo logs must be separately backed up and restored for recovery operations.

Note that Oracle's best practice for database backup and recovery is to use Recovery Manager (RMAN) only, not storage snapshots, and is integrated with Zero Data Loss Recovery Appliance (ZDLRA) to offer the highest levels of database protection.

Related Topics

- IT Infrastructure Engineered Systems Zero Data Loss Recovery Appliance

Desupported Parameters in Oracle Database 23ai

As part of your upgrade plan, review the initialization parameters that are not supported starting with this Oracle Database release.

- EXTERNAL_NAME Parameter in SYS_CONTEXT USERENV Desupported
  The `EXTERNAL_NAME` parameter in `SYS_CONTEXT USERENV` is desupported in Oracle Database 23ai
- Service Attribute Value SESSION_STATE_CONSISTENCY = STATIC Desupported
  The service attribute values `FAILOVER_TYPE = TRANSACTION` with `SESSION_STATE_CONSISTENCY = STATIC` are no longer a supported service attribute combination.
- Oracle Label Security Parameters and Functions Desupported
  Oracle Label Security (OLS) parameters and functions that were deprecated in Oracle Database 12c are desupported in Oracle Database 23ai.
- ENCRYPTION_WALLET_LOCATION Parameter Desupported
  Starting with Oracle Database 23ai, the parameter `ENCRYPTION_WALLET_LOCATION` is desupported.
- ADD_SSLV3_TO_DEFAULT SQLNET.ORA Parameter (and SSLv3) Desupported
  Starting with Oracle Database 23ai, the Secure Socket Layer v3 protocol (SSLv3) is no longer supported for database server-client connections, and the `sqlnet.ora` parameter `ADD_SSLV3_TO_DEFAULT` has been removed.
- Desupport of OPTIMIZER_SECURE_VIEW_MERGING Parameter
  Oracle is desupporting the initialization parameter `OPTIMIZER_SECURE_VIEW_MERGING` in Oracle Database 23ai

EXTERNAL_NAME Parameter in SYS_CONTEXT USERENV Desupported

The `EXTERNAL_NAME` parameter in `SYS_CONTEXT USERENV` is desupported in Oracle Database 23ai

Date: April 2023

The `EXTERNAL_NAME` parameter was deprecated in Oracle Database 10g. Two newer parameters, `AUTHENTICATED_IDENTITY` and `ENTERPRISE_IDENTITY`, offer more information. `EXTERNAL_NAME` is removed in Oracle Database 23ai.

Service Attribute Value SESSION_STATE_CONSISTENCY = STATIC Desupported

The service attribute values `FAILOVER_TYPE = TRANSACTION` with `SESSION_STATE_CONSISTENCY = STATIC` are no longer a supported service attribute combination.

Date: April 2023

In previous releases, you could use the service parameter `SESSION_STATE_CONSISTENCY` to manage session state automatically using Application Continuity by setting `SESSION_STATE_CONSISTENCY` to `DYNAMIC` or `STATIC`. However, starting with Oracle Database 23ai, you can no longer use the `STATIC` option. Instead, use one of the following failover options:

- `FAILOVER_TYPE = AUTO` with `SESSION_STATE_CONSISTENCY = AUTO`
- `FAILOVER_TYPE = TRANSACTION` with `SESSION_STATE_CONSISTENCY = DYNAMIC`

These configurations enforce session state tracking in Oracle Database, ensuring that session state is preserved at session migration and session failover.

Note that starting with Oracle Database 21c, you may also wish to set the `RESET_STATE` attribute to clear your session state set by applications in request at the end of the request. For more information, see `RESET_STATE`. For planned maintenance, Oracle recommends that you drain requests from Oracle Database connection pools in combination with Application Continuity for those requests that do not complete.

Related Topics

- Session State Consistency

Oracle Label Security Parameters and Functions Desupported

Oracle Label Security (OLS) parameters and functions that were deprecated in Oracle Database 12c are desupported in Oracle Database 23ai.

Date: April 2023

These previously deprecated OLS parameters and functions have alternate parameters that can be used.

The following OLS functions and parameters are desupported:

`Least_UBOUND`. Use `OLS_GREATEST_LBOUND` instead.
`LUBD`. Use `OLS_GLBD` instead.
`DOMINATES`. Use the `OLS_DOMINATES` standalone function instead.
`DOM`. Use the `OLS_STRICTLY_DOMINATES` standalone function instead.
`STRICTLY_DOMINATES`. Use the `OLS_STRICTLY_DOMINATES` standalone function instead.
`S_DOM`. Use the `OLS_STRICTLY_DOMINATES` standalone function instead.
`DOMINATED_BY`. Use the `OLS_DOMINATED_BY` standalone function instead.
`DOM_BY`. Use the `OLS_DOMINATED_BY` standalone function instead.
`STRICTLY_DOMINATED_BY`. Use the OLS_STRICTLY_DOMINATED_BY standalone function instead.
`S_DOM_BY`. Use the `OLS_STRICTLY_DOMINATED_BY` standalone function instead.

Also, the OLS standalone function `SA_UTL.DOMINATES` boolean datatype is desupported. The `SA_UTL.DOMINATES` function that uses the `NUMBER` datatype is not deprecated.

For reference about alternatives see: Oracle Label Security Administrator's Guide Deprecated Features

ENCRYPTION_WALLET_LOCATION Parameter Desupported

Starting with Oracle Database 23ai, the parameter `ENCRYPTION_WALLET_LOCATION` is desupported.

Date: April 2023

To store and retrieve the TDE wallet, use the `WALLET_ROOT` structure (introduced with Oracle Database 18c).

> **Note:**
>
> If Transparent Data Encryption (TDE) is enabled, but WALLET_ROOT is not configured, then you will be blocked from upgrading to Oracle Database 23ai. This block for upgrades of databases using TDE is to prevent the possibility of not being able to open the database after the upgrade.

ADD_SSLV3_TO_DEFAULT SQLNET.ORA Parameter (and SSLv3) Desupported

Starting with Oracle Database 23ai, the Secure Socket Layer v3 protocol (SSLv3) is no longer supported for database server-client connections, and the `sqlnet.ora` parameter `ADD_SSLV3_TO_DEFAULT` has been removed.

Date: April 2023

SSLv3 is a much less secure protocol to secure the database server-to-client connection. Instead of using SSLv3, allow the database server and client to negotiate the most secure protocol that is common between the server and the client. Oracle Database 23ai provides TLS 1.2 and TLS 1.3 protocols for certificate-based network encryption.

Desupport of OPTIMIZER_SECURE_VIEW_MERGING Parameter

Oracle is desupporting the initialization parameter `OPTIMIZER_SECURE_VIEW_MERGING` in Oracle Database 23ai

April 2024

The `init.ora` parameter `OPTIMIZER_SECURE_VIEW_MERGING` enabled the optimizer to use view merging without performing the checks that would otherwise be performed to ensure that view merging does not violate any security intentions of the view creator. There is no replacement for this parameter. If you need to use this parameter, then contact Oracle Support.

Deprecated Features in Oracle Database 23ai

As part of your upgrade plan, review the features that are deprecated in this Oracle Database release, and review alternatives for your application strategies.

- PROXY_ONLY_CONNECT Deprecation
  The ability to set a schema to `PROXY_ONLY_CONNECT` is deprecated in Oracle Database 23ai.

- Oracle Database 11g SHA-1 Verifier Deprecated
  Starting with Oracle Database 23ai, the SHA-1 verifier introduced with Oracle Database 11g is deprecated.
- RADIUS API Based on RFC-2138 is Deprecated
  Starting with Oracle Database 23ai, the older RADIUS API that is based on Request for Comments (RFC) 2138 is deprecated.
- Enterprise User Security (EUS) Deprecated
  Enterprise User Security (EUS) is deprecated with Oracle Database 23ai.
- Auto-Login Wallet Version 6 Deprecated
  Oracle has introduced a new auto-login wallet version (7) with Oracle Database 23ai. Version 6 of the Oracle local auto-login wallet is deprecated.
- WALLET_LOCATION Parameter Deprecated
  The parameter `WALLET_LOCATION` is deprecated for use with Oracle Database 23ai for the Oracle Database server. It is not deprecated for use with the Oracle Database client or listener.
- RAS Mid-Tier Session Support for Fusion Middleware Deprecated
  Oracle Real Application Security (RAS) mid-tier master session support for Fusion Middleware application session service is deprecated with Oracle Database 23ai
- Oracle Data Provider for .NET, Unmanaged Driver Deprecation
  Oracle Data Provider for .NET (ODP.NET), Unmanaged Driver is deprecated in Oracle Database 23ai.
- GOST and SEED Algorithms Deprecation
  Starting with Oracle Database 23ai, the Transparent Data Encryption (TDE) decryption libraries for the GOST and SEED algorithms are deprecated, and encryption to GOST and SEED are desupported.
- Oracle Persistent Memory Deprecation
  Oracle Persistent Memory Database (PMEM) is deprecated as of Oracle Database 23ai due to Intel discontinuing Optane Persistent Memory hardware.
- PMEM Support in Oracle Memory Speed File System Deprecation
  Oracle Memory Speed (OMS) File System is deprecated in Oracle Database 23ai for use with Intel Optane Persistent Memory.
- Service Name with Partial DN Matching and Server-only Certificate Check Deprecation
  Starting with Oracle Database 23ai, matching (partial and full) is not limited to the database server certificate. The listener certificate is also checked, and the `SERVICE_NAME` parameter is ignored for partial DN matching.
- Deprecation of the mkstore Command-Line Utility
  The `mkstore` wallet management command line tool is deprecated with Oracle Database 23ai, and can be removed in a future release.
- Network Data Model (NDM) XML API Deprecation
  The Network Data Model (NDM) XML API used by the NDM feature of Oracle Spatial is deprecated in Oracle Database Release 23ai.
- Two Subprograms in SDO_GEOR_ADMIN Package Deprecation
  The GeoRaster subprograms `SDO_GEOR_ADMIN.isUpgradeNeeded` and `SDO_GEOR_ADMIN.upgradeGeoRaster` in `SDO_GEOR_ADMIN` package, which are used to maintain the GeoRaster objects in the database, are deprecated in Oracle Database Release 23ai.

- SDO_GEOR.importFrom and SDO_GEOR.exportTo subprograms in SDO_GEOR Package Deprecations
  The GeoRaster subprograms `SDO_GEOR.importFrom` and `SDO_GEOR.exportTo` in the `SDO_GEOR` package are deprecated in Oracle Database Release 23ai.
- WAIT Option of Oracle Data Guard SWITCHOVER Command
  Starting with Oracle Database 23ai, the `WAIT` option of the Oracle Data Guard `SWITCHOVER` command is deprecated.
- DBMS_RESULT_CACHE Function Name Deprecations
  Oracle is changing the names of several DBMS_RESULT_CACHE function names in Oracle Database 23ai.
- Oracle ACFS Snapshot Remastering Deprecation
  Starting with Oracle Database Release 23ai, the Snapshot Remastering feature of Oracle Advanced Cluster File System (ACFS) is deprecated.
- Oracle ACFS Compression Deprecation
  The ACFS compression feature of Oracle Advanced Cluster File System (ACFS) is deprecated in Oracle Database 23ai.
- Oracle Virtual Directory with Real Application Security Deprecation
  The use of Oracle Virtual Directory with Oracle Real Application Security is deprecated with Oracle Database 23ai.
- BIG_IO Attribute in Oracle Text Deprecation
  The `BIG_IO` attribute of the `CONTEXT` indextype is deprecated with Oracle Database 23ai, and can be disabled or removed in a future release.
- ASYNCHRONOUS Attribute in Oracle Text Deprecation
  The `ASYNCHRONOUS_UPDATE` setting of the `CONTEXT` indextype is deprecated in Oracle Database 23ai, and can be ignored or removed in a future release.
- Oracle Text CTXCAT Indextype Deprecation
  The Oracle Text indextype `CTXCAT` is deprecated with Oracle Database 23ai. The indextype itself, and it's operator `CTXCAT`, can be removed in a future release.
- Oracle XML DB Repository Deprecation
  The Oracle XML DB Repository is deprecated with Oracle Database 23ai.
- DBMS_XMLGEN Deprecation
  The PL/SQL package `DBMS_XMLGEN` is deprecated in Oracle Database 23ai.
- DBMS_XMLSTORE Deprecation
  The PL/SQL package `DBMS_XMLSTORE` is deprecated in Oracle Database 23ai.
- Deprecation of Unstructured XML Indexes
  Unstructured XML indexes are deprecated in Oracle Database 23ai.
- DBMS_HANG_MANAGER Package Deprecation
  The `DBMS_HANG_MANAGER` package is deprecated in Oracle Database 23ai. Use `DBMS_BLOCKER_RESOLVER` instead.
- Zero Downtime Upgrade (ZDU) Deprecation
  The Zero Downtime Upgrade (ZDU) feature of Oracle Fleet Patching and Provisioning (FPP) is deprecated in Oracle Database 23ai.
- Highly Available Grid Naming Service (GNS) Deprecation
  The Highly Available Grid Naming Service feature of Grid Naming Service (GNS) in Oracle Grid Infrastructure is deprecated in Oracle Database 23ai.
- Traditional Auditing Packages and Functions Deprecated
  Traditional auditing packages and functions are deprecated in Oracle Database 23ai.

- MDSYS-Owned RDF Graph Networks Deprecated
  Creation of RDF graph networks in the MDSYS schema is deprecated. Oracle recommends that you create RDF graph networks in a user schema, which was enabled in Oracle Database 19c.
- TREAT (expr AS JSON) Deprecated
  The operator `TREAT (expr AS JSON)` is deprecated, because SQL data type JSON is now available.
- Deprecation of Out-of-Place Patching with Opatch and OPatchAuto
  The use of OPatch and OPatchAuto for out-of-place patching continues to be deprecated.
- RCONFIG Command-Line Interface Deprecation
  Starting with Oracle Database 23ai, the `RCONFIG` utility is deprecated.
- UPDATE_SDATA API Deprecation
  The `UPDATE_SDATA` API in Oracle Text is deprecated in Oracle Database 23ai.
- Oracle JDBC-OCI Thick Driver Deprecation
  The Oracle JDBC Oracle Call Interface (OCI) Type 2 client driver (also known as a "thick" driver) is deprecated in Oracle Database 23ai.
- Deprecation of SQLJ
  Starting with Oracle Database 23ai, the SQLJ method of embedding SQL statements in Java code is deprecated.
- Oracle JDBC Proprietary BLOB/CLOB Open and Close Methods Deprecation
  The Oracle JDBC methods `open()`, `close()`, and `isClosed()` in `OracleBlob`, `OracleClob`, and `OracleBfile` are deprecated for removal in Oracle Database 23ai.

PROXY_ONLY_CONNECT Deprecation

The ability to set a schema to `PROXY_ONLY_CONNECT` is deprecated in Oracle Database 23ai.

Date: April 2023

Setting a schema to be `PROXY_ONLY_CONNECT` is deprecated in this release, and can be desupported in a future release. Oracle recommends that you use `SCHEMA ONLY` user accounts instead of `PROXY_ONLY_CONNECT`. Schema-Only Accounts don't have passwords associated with the schema, so you can't directly log in to the schema, and passwords don't need to be rotated.

Oracle Database 11g SHA-1 Verifier Deprecated

Starting with Oracle Database 23ai, the SHA-1 verifier introduced with Oracle Database 11g is deprecated.

Date: April 2023

The salted multi-round SHA-512 password hash (also known as "verifier") introduced with Oracle Database 12c provides enhanced security for your password. If 11g verifiers (`11G`) are still being used in your database, then Oracle recommends resetting them so they can be upgraded to the 12c (`12C`) de-optimized `PBKDF2`-based verifier.

RADIUS API Based on RFC-2138 is Deprecated

Starting with Oracle Database 23ai, the older RADIUS API that is based on Request for Comments (RFC) 2138 is deprecated.

Oracle Database 23ai introduces an updated RADIUS API based on RFC 6613 and RFC 6614. Oracle recommends that you start planning on migrating to use the new RADIUS API as soon as possible. The new API is enabled by default. These parameters associated with the older RADIUS API are also deprecated: `SQLNET.RADIUS_ALTERNATE`, `SQLNET.RADIUS_ALTERNATE_PORT`, `SQLNET.RADIUS_AUTHENTICATION`, and `SQLNET.RADIUS_AUTHENTICATION_PORT`. Refer to the Radius API documentation for information on changing the default to use the older RADIUS API.

Enterprise User Security (EUS) Deprecated

Enterprise User Security (EUS) is deprecated with Oracle Database 23ai.

Oracle recommends that you migrate to using Centrally Managed Users (CMU). This feature enables you to directly connect with Microsoft Active Directory without an intervening directory service for enterprise user authentication and authorization to the database. If your Oracle Database is in the cloud, you can also choose to move to one of the newer integrations with a cloud identity provider.

Auto-Login Wallet Version 6 Deprecated

Oracle has introduced a new auto-login wallet version (7) with Oracle Database 23ai. Version 6 of the Oracle local auto-login wallet is deprecated.

Date: April 2023

You can update your local auto-login wallet by modifying it with `orapki`.

WALLET_LOCATION Parameter Deprecated

The parameter `WALLET_LOCATION` is deprecated for use with Oracle Database 23ai for the Oracle Database server. It is not deprecated for use with the Oracle Database client or listener.

Date: April 2023

For Oracle Database server, Oracle recommends that you use the `WALLET_ROOT` system parameter instead of using `WALLET_LOCATION`.

RAS Mid-Tier Session Support for Fusion Middleware Deprecated

Oracle Real Application Security (RAS) mid-tier master session support for Fusion Middleware application session service is deprecated with Oracle Database 23ai

Date: April 2023

This filter created the RAS session automatically for the FMW application server. Starting with Oracle Database 23ai, you now need to write the code to create the RAS session in the future, just as you would be required to do for any other application server.

Oracle Data Provider for .NET, Unmanaged Driver Deprecation

Oracle Data Provider for .NET (ODP.NET), Unmanaged Driver is deprecated in Oracle Database 23ai.

Date: April 2023

ODP.NET provides ADO.NET-based data access to Oracle Database. There are two primary Oracle data access drivers for Microsoft .NET Framework: ODP.NET, Managed Driver and

ODP.NET, Unmanaged Driver. In Oracle Database 23ai, ODP.NET, Managed Driver supports all major features available in ODP.NET, Unmanaged Driver with the same application programming interfaces and configuration settings. Code migration from unmanaged ODP.NET to managed ODP.NET is straightforward for the vast majority of existing .NET applications. Oracle recommends that you migrate existing unmanaged ODP.NET applications to ODP.NET, Managed Driver. The ODP.NET, Unmanaged Driver can be desupported in a future release.

ODP.NET, Managed Driver is a more compact and simpler install that can be consumed via NuGet packaging. Unmanaged ODP.NET is not available as a NuGet package. It is easier to manage multiple managed ODP.NET deployments on the same machine than multiple unmanaged ODP.NET deployments. These advantages make managed ODP.NET preferable for customer use over unmanaged ODP.NET. Because unmanaged ODP.NET no longer has any advantages over managed ODP.NET, Oracle has chosen to deprecate unmanaged ODP.NET.

GOST and SEED Algorithms Deprecation

Starting with Oracle Database 23ai, the Transparent Data Encryption (TDE) decryption libraries for the GOST and SEED algorithms are deprecated, and encryption to GOST and SEED are desupported.

Date: April 2023

GOST 28147-89 has been deprecated by the Russian government, and SEED has been deprecated by the South Korean government. If you need South Korean government-approved TDE cryptography, then use ARIA instead. If you are using GOST 28147-89, then you must decrypt and encrypt with another supported TDE algorithm. The decryption algorithms for GOST 28147-89 and SEED are included in Oracle Database 23ai, but are deprecated, and the GOST encryption algorithm is desupported with Oracle Database 23ai. If you are using GOST or SEED for TDE encryption, then Oracle recommends that you online re-key to another algorithm before upgrading to Oracle Database 23ai. However, with the exception of the HP Itanium platform, the GOST and SEED decryption libraries are available with Oracle Database 23ai, so you can also decrypt after upgrading.

Oracle Persistent Memory Deprecation

Oracle Persistent Memory Database (PMEM) is deprecated as of Oracle Database 23ai due to Intel discontinuing Optane Persistent Memory hardware.

Date: April 2023

Intel has announced they will discontinue the Optane Persistent Memory product. Therefore, Oracle Persistent Memory Database is being deprecated.

PMEM Support in Oracle Memory Speed File System Deprecation

Oracle Memory Speed (OMS) File System is deprecated in Oracle Database 23ai for use with Intel Optane Persistent Memory.

Date: April 2023

Intel has announced they will discontinue the Optane Persistent Memory product. Therefore, use of Oracle Memory Speed (OMS) File System is being deprecated.

Service Name with Partial DN Matching and Server-only Certificate Check Deprecation

Starting with Oracle Database 23ai, matching (partial and full) is not limited to the database server certificate. The listener certificate is also checked, and the `SERVICE_NAME` parameter is ignored for partial DN matching.

Date: April 2023

Server-side certificate verification through distinguished name (DN) is changed as follows: Both the listener certificate and the database server certificate are checked. In earlier Oracle Database releases, only the database server certificate was checked. In most production cases, the same certificate is used by the listener and the database. In cases where different certificates are used, DN matching can require new certificates to allow partial DN matching on SAN or hostname certificate information. In addition to checking the listener certificate, when using partial DN matching is used, the `SERVICE_NAME` parameter will be ignored Only the hostname connect string parameter will be checked against the certificate common name (CN) and subject alternate name (SAN) fields. To revert to the behavior in earlier releases (using the service name in addition to hostname, and only checking the database server certificate), set the new parameter: `SSL_ALLOW_WEAK_DN_MATCH=TRUE`. The default is `FALSE`.

Deprecation of the mkstore Command-Line Utility

The `mkstore` wallet management command line tool is deprecated with Oracle Database 23ai, and can be removed in a future release.

Date: April 2023

To manage wallets, Oracle recommends that you use the `orapki` command line tool.

Network Data Model (NDM) XML API Deprecation

The Network Data Model (NDM) XML API used by the NDM feature of Oracle Spatial is deprecated in Oracle Database Release 23ai.

Date: April 2023

To efficiently and securely implement your customized network analysis, Oracle Spatial is deprecating the NDM XML API. Instead, use the Load On Demand (LOD) API. The LOD API provides customization interfaces that allow you to easily and securely implement your customization. In addition, you can also use the NDM Contraction Hierarchies REST API for high performance network analysis.

Two Subprograms in SDO_GEOR_ADMIN Package Deprecation

The GeoRaster subprograms `SDO_GEOR_ADMIN.isUpgradeNeeded` and `SDO_GEOR_ADMIN.upgradeGeoRaster` in `SDO_GEOR_ADMIN` package, which are used to maintain the GeoRaster objects in the database, are deprecated in Oracle Database Release 23ai.

Date: April 2023

As the size of the GeoRaster database increases, the performance of the `SDO_GEOR_ADMIN.isUpgradeNeeded` and `SDO_GEOR_ADMIN.upgradeGeoRaster` subprograms are not efficient. Instead, Oracle recommends that you use alternative subprograms, such as `SDO_GEOR_ADMIN.checkSysdataEntries`, `SDO_GEOR_ADMIN.maintainSysdataEntries` and a few

other subprograms to maintain the GeoRaster objects and their system data in Oracle Database.

Detailed instructions on how to run the alternative subprograms to maintain GeoRaster databases are available in *Oracle Spatial Spatial GeoRaster Developer's Guide*.

SDO_GEOR.importFrom and SDO_GEOR.exportTo subprograms in SDO_GEOR Package Deprecations

The GeoRaster subprograms `SDO_GEOR.importFrom` and `SDO_GEOR.exportTo` in the `SDO_GEOR` package are deprecated in Oracle Database Release 23ai.

Date: April 2023

The `SDO_GEOR.importFrom` and `SDO_GEOR.exportTo` subprograms support very small images stored in a few general image file formats. They are limited by many usage restrictions, by low performance, and they are not supported in Oracle Autonomous Database. To load and export image and raster files, Oracle recommends that you replace these deprecated subprograms with alternative options, such as the SDO_GEOR_GDAL.translate subprogram, GDAL, or the GDAL-Based GeoRaster ETL . These options support various geospatial images in more than 100 formats without many restrictions, and have much greater performance and scalability.

WAIT Option of Oracle Data Guard SWITCHOVER Command

Starting with Oracle Database 23ai, the `WAIT` option of the Oracle Data Guard `SWITCHOVER` command is deprecated.

Date: April 2023

In Oracle Database 21c, Oracle introduced a new Oracle Data Guard property, `DrainTimeout`, which overrides the default for `drain_timeout` of database services registered in Oracle Clusterware when a `SWITCHOVER` is issued. The Oracle Data Guard `WAIT` option in the `SWITCHOVER` command is now redundant. Oracle recommends the new `DrainTimeout` property.

DBMS_RESULT_CACHE Function Name Deprecations

Oracle is changing the names of several DBMS_RESULT_CACHE function names in Oracle Database 23ai.

Date: April 2023

The following functions and procedures are deprecated:

- `BLACK_LIST` function. Use `BLOCK_LIST` function.
- BLACK_LIST_ADD procedure. Use BLOCK_LIST_ADD procedure.
- BLACK_LIST_CLEAR procedure. Use BLOCK_LIST_CLEAR procedure
- BLACK_LIST_REMOVE procedure. Use BLOCK_LIST_REMOVE procedure
- OBJECT_BLACK_LIST function. Use OBJECT_BLOCK_LIST function
- OBJECT_BLACK_LIST_ADD procedure. Use OBJECT_BLOCK_LIST_ADD procedure.
- OBJECT_BLACK_LIST_CLEAR procedure. Use OBJECT BLOCK LIST_CLEAR procedure.
- OBJECT_BLACK_LIST_REMOVE procedure. Use OBJECT_BLOCK_LIST_REMOVE procedure.

Oracle ACFS Snapshot Remastering Deprecation

Starting with Oracle Database Release 23ai, the Snapshot Remastering feature of Oracle Advanced Cluster File System (ACFS) is deprecated.

Date: April 2023

This feature can be desupported in a future release without replacement. Desupporting features with limited adoption enables Oracle to focus on improving core scaling, availability, and manageability.

Oracle ACFS Compression Deprecation

The ACFS compression feature of Oracle Advanced Cluster File System (ACFS) is deprecated in Oracle Database 23ai.

Date: April 2023

This feature can be desupported in a future release without replacement. Desupporting features with limited adoption enables Oracle to focus on improving core scaling, availability, and manageability.

Oracle Virtual Directory with Real Application Security Deprecation

The use of Oracle Virtual Directory with Oracle Real Application Security is deprecated with Oracle Database 23ai.

Date: July 2023

Using OVD with Oracle Real Application Security is deprecated, because OVD is no longer updated as a separate product

BIG_IO Attribute in Oracle Text Deprecation

The `BIG_IO` attribute of the `CONTEXT` indextype is deprecated with Oracle Database 23ai, and can be disabled or removed in a future release.

Date: July 2023

Oracle recommends that you allow this value to be set to its default value of `N`. `BIG_IO` was introduced to reduce the cost of seeks when index postings exceeded 4KB in length. However, the internal code is relatively inefficient, and the attribute cannot be combined with newer index options. Seek cost is much less relevant for solid state disks or non-volatile memory devices (NVMe), and seek cost is irrelevant when postings are cached. This setting is therefore of little benefit for most indexes.

ASYNCHRONOUS Attribute in Oracle Text Deprecation

The `ASYNCHRONOUS_UPDATE` setting of the `CONTEXT` indextype is deprecated in Oracle Database 23ai, and can be ignored or removed in a future release.

Date: July 2023

Oracle can ignore or remove this attribute in a future release. Oracle recommends that you allow this value to be set to its default value, `SYNCHRONOUS_UPDATE`. To avoid unexpected loss of results during updates, use `SYNC (ON COMMIT)` or `SYNC(EVERY [time-period])` with a short time period.

The `ASYNCHRONOUS_UPDATE` setting was introduced as a workaround for the fact that updates are implemented as "delete followed by insert," and that deletes are immediate (on commit), while inserts are only performed during an index sync. However, this setting is incompatible with several other index options. Oracle recommends that you discontinue its use.

Oracle Text CTXCAT Indextype Deprecation

The Oracle Text indextype `CTXCAT` is deprecated with Oracle Database 23ai. The indextype itself, and it's operator `CTXCAT`, can be removed in a future release.

Date: July 2023

Both `CTXCAT` and the use of `CTXCAT` grammar as an alternative grammar for `CONTEXT` queries is deprecated. Instead, Oracle recommends that you use the `CONTEXT` indextype, which can provide all the same functionality, except that it is not transactional. Near-transactional behavior in `CONTEXT` can be achieved by using `SYNC(ON COMMIT)` or, preferably, `SYNC(EVERY [time-period])` with a short time period.

`CTXCAT` was introduced when indexes were typically a few megabytes in size. Modern, large indexes, can be difficult to manage with `CTXCAT`. The addition of index sets to `CTXCAT` can be achieved more effectively by the use of `FILTER BY` and `ORDER BY` columns, or `SDATA`, or both, in the `CONTEXT` indextype. `CTXCAT` is therefore rarely an appropriate choice. Oracle recommends that you choose the more efficient `CONTEXT` indextype.

Oracle XML DB Repository Deprecation

The Oracle XML DB Repository is deprecated with Oracle Database 23ai.

Date: July 2023

Oracle recommends that you replace any functionality used in XML DB Repository with alternative technologies. As a result of this deprecation, all XML DB Repository interfaces (for example Repository-specific Java classes `oracle.xdb.servlet`, `oracle.xdb.event`, and `oracle.xdb.spi`) are consequently deprecated as well.

DBMS_XMLGEN Deprecation

The PL/SQL package `DBMS_XMLGEN` is deprecated in Oracle Database 23ai.

Date: July 2023

`DBMS_XMLGEN` is a non-standard Oracle-proprietary package that is provided to generate and convert XML documents from SQL queries or with PL/SQL. This package is deprecated, and can be desupported in a future release. Oracle recommends that you use SQL/XML operators to generate XML from relational columns instead. Using ANSI SQL/XML operators for any generation and modification of XML documents provides a standardized and future-proof way to work with XML documents.

DBMS_XMLSTORE Deprecation

The PL/SQL package `DBMS_XMLSTORE` is deprecated in Oracle Database 23ai.

Date: July 2023

`DBMS_XMLSTORE` is a non-standard Oracle-proprietary package that enables you to store and manipulate XML data in Oracle Database. This package is deprecated, and can be desupported in a future release. Oracle recommends that you use regular SQL DML and with

standard XQuery and SQL/XML to store and manage XML data. Using standard functionality provides future-proof way to store and manipulate XML data.

Deprecation of Unstructured XML Indexes

Unstructured XML indexes are deprecated in Oracle Database 23ai.

Date: July 2023

Unstructured XML indexes are deprecated and superseded by XML search indexes. Oracle recommends that you use XML search indexes or structured XML indexes.

DBMS_HANG_MANAGER Package Deprecation

The `DBMS_HANG_MANAGER` package is deprecated in Oracle Database 23ai. Use `DBMS_BLOCKER_RESOLVER` instead.

Date: July 2023

The `DBMS_HANG_MANAGER` package provides a method of changing some configuration parameters and constraints to address session issues. This package is being replaced with `DBMS_BLOCKER_RESOLVER`. `DBMS_HANG_MANAGER` can be removed in a future release.

Zero Downtime Upgrade (ZDU) Deprecation

The Zero Downtime Upgrade (ZDU) feature of Oracle Fleet Patching and Provisioning (FPP) is deprecated in Oracle Database 23ai.

Date: July 2023

ZDU uses either Transient Logical Standby or Oracle GoldenGate classic. However, Transient Logical Standby does not support the latest data types, and the Oracle GoldenGate classic implementation is deprecated. Instead of ZDU, Oracle recommends that you use FPP in combination with the `DBMS_ROLLING` PL/SQL package: You can use FPP for the provisioning and deployment of new Oracle homes, and you can use `DBMS_ROLLING` as a streamlined method of performing rolling upgrades.

The `DBMS_ROLLING` package is closely integrated with Application Continuity to help to minimize disruptions in the database tier, and mask rolling patching activities.

Related Topics

- Using DBMS_ROLLING to Perform a Rolling Upgrade

Highly Available Grid Naming Service (GNS) Deprecation

The Highly Available Grid Naming Service feature of Grid Naming Service (GNS) in Oracle Grid Infrastructure is deprecated in Oracle Database 23ai.

Date: December 2023

The highly-available GNS provides the ability to run multiple GNS instances in a multi-cluster environment with different roles. This feature is being deprecated. There is no replacement.

Traditional Auditing Packages and Functions Deprecated

Traditional auditing packages and functions are deprecated in Oracle Database 23ai.

Date: September 2023

With the desupport of traditional auditing, the PL/SQL packages and functions associated with traditional auditing are deprecated, This deprecation includes the packages and functions `INIT_CLEANUP`, `DEINIT_CLEANUP`, and `IS_CLEANUP_INITIALIZED`. While these packages or functions continue to operate in Oracle Database 23ai, you can neither add to or modify traditional auditing configurations.

MDSYS-Owned RDF Graph Networks Deprecated

Creation of RDF graph networks in the MDSYS schema is deprecated. Oracle recommends that you create RDF graph networks in a user schema, which was enabled in Oracle Database 19c.

Date: November 2023

An RDF graph network is a logical structure a developer creates to contain multiple RDF graphs. Starting with Oracle Database 19c, RDF graph networks can be created in a user schema. It is simpler for developers to create RDF graph networks in a user schema, and this method makes it easier to share graphs with other user schemas. Creating RDF graph networks in user schemas is also more secure, as developers can create RDF graph networks without requiring DBA privileges.

TREAT (*expr* AS JSON) Deprecated

The operator `TREAT (expr AS JSON)` is deprecated, because SQL data type JSON is now available.

Date: November 2023

The `TREAT(expr as JSON)` functionality is no longer needed, as this function is replaced with the JSON constructor. For this reason, Oracle is deprecating the `TREAT AS JSON` operator in Oracle Database 23ai.

Deprecation of Out-of-Place Patching with Opatch and OPatchAuto

The use of OPatch and OPatchAuto for out-of-place patching continues to be deprecated.

Date: December 2023

For patching in Oracle Database 23ai, Oracle recommends that you use out-of-place patching using Gold images to apply quarterly release updates (RUs). This deprecation does not affect in-place patching. If you want to perform in-place patching, then OPatch and OPatchAuto continue to be available for this purpose. The Oracle Database 23ai Free release is not supported for patching with RUs.

RCONFIG Command-Line Interface Deprecation

Starting with Oracle Database 23ai, the `RCONFIG` utility is deprecated.

Date: January 2024

Oracle recommends that you use Database Configuration Assistant (DBCA). Beginning with Oracle Enterprise Manager 13c Release 5 Update 23 (13.5.0.23), converting a single-instance database to Oracle RAC through Oracle Enterprise Manager using the `RCONFIG` utility is no longer supported.

Related Topics

- Converting Single-Instance Oracle Databases to Oracle RAC and Oracle RAC One Node

UPDATE_SDATA API Deprecation

The `UPDATE_SDATA` API in Oracle Text is deprecated in Oracle Database 23ai.

Date: February 2024

Instead of modifying the index, Oracle recommends that you update the underlying data.

Oracle JDBC-OCI Thick Driver Deprecation

The Oracle JDBC Oracle Call Interface (OCI) Type 2 client driver (also known as a "thick" driver) is deprecated in Oracle Database 23ai.

Date: May 2024

Most Java applications that use Open Database Connectivity (ODBC) with Oracle JDBC drivers use the Thin driver. To enable Oracle to allocate resources to better address customer requirements, Oracle is deprecating the JDBC-OCI driver.

Deprecation of SQLJ

Starting with Oracle Database 23ai, the SQLJ method of embedding SQL statements in Java code is deprecated.

Date: May 2024

In place of SQLJ, Oracle recommends that you directly use the Java Database Connectivity (JDBC) APIs (dynamic SQL, prepared statements or PL/SQL blocks).

Oracle JDBC Proprietary BLOB/CLOB Open and Close Methods Deprecation

The Oracle JDBC methods `open()`, `close()`, and `isClosed()` in `OracleBlob`, `OracleClob`, and `OracleBfile` are deprecated for removal in Oracle Database 23ai.

Date: July 2024

Oracle is deprecating these methods, which are replaced with `openLob()`, `closeLob()` and `isClosedLob()`. The method `close()` conflicts with the interface type `java.lang.Autocloseable`. Removing the proprietary method `close()` makes it possible for `java.lang OracleBlob`, `java.lang.OracleClob`, and `java.lang.OracleBfile` to extend the Autocloseable interface at some future time. The `open()` and `isClosed()` methods will be removed and replaced in a future release to maintain rational names for these methods.

Deprecated Views in Oracle Database 23ai

As part of your upgrade plan, review the views that are deprecated starting with this Oracle Database release.

- V$DATABASE.FS_FAILOVER Columns in V$DATABASE View Deprecated
  Starting with Oracle Database 23ai, all `V$DATABASE.FS_FAILOVER` columns in the `V$DATABASE` view are deprecated. Use the `V$FAST_START_FAILOVER_CONFIG` view.
- V$PQ_SLAVE View Deprecation
  Oracle is replacing the view `V$PQ_SLAVE` with `V$PX_SERVER` in Oracle Database 23ai.

- V$FS_FAILOVER_STATS View Deprecation
  The view V$FS_FAILOVER_STATS is deprecated and replaced by the view V$DG_BROKER_ROLE_CHANGE in Oracle Database 23ai.
- DBA_HANG_MANAGER_PARAMETERS Data Dictionary View Deprecation
  The DBA_HANG_MANAGER_PARAMETERS static Data Dictionary view is deprecated with Oracle Database 23ai. Use DBA_BLOCKER_RESOLVER_PARAMETERS instead.
- V$RECOVERY_SLAVE View Deprecation
  Oracle is replacing the view V$RECOVERY_SLAVE with V$RECOVERY_WORKER in Oracle Database 23ai.

V$DATABASE.FS_FAILOVER Columns in V$DATABASE View Deprecated

Starting with Oracle Database 23ai, all V$DATABASE.FS_FAILOVER columns in the V$DATABASE view are deprecated. Use the V$FAST_START_FAILOVER_CONFIG view.

Date: April 2023

Oracle is introducing the V$FAST_START_FAILOVER_CONFIG view, which contains configuration details about the Oracle Data Guard fast-start failover feature. As a result, the FS_FAILOVER_* columns in the V$DATABASE view are deprecated. Instead, use the corresponding columns in the V$FAST_START_FAILOVER_CONFIG view.

The column replacements are as follows:

- V$DATABASE.FS_FAILOVER_MODE: Replace with V$FAST_START_FAILOVER_CONFIG.FAST_START_FAILOVER_MODE
- V$DATABASE.FS_FAILOVER_STATUS: Replace with V$FAST_START_FAILOVER_CONFIG.STATUS
- V$DATABASE.FS_FAILOVER_CURRENT_TARGET: Replace with V$FAST_START_FAILOVER_CONFIG.CURRENT_TARGET
- V$DATABASE.FS_FAILOVER_THRESHOLD: Replace with V$FAST_START_FAILOVER_CONFIG.THRESHOLD
- V$DATABASE.FS_FAILOVER_OBSERVER_PRESENT: Replace with V$FAST_START_FAILOVER_CONFIG.OBSERVER_PRESENT
- V$DATABASE.FS_FAILOVER_OBSERVER_HOST: Replace with V$FAST_START_FAILOVER_CONFIG.OBSERVER_HOST

V$PQ_SLAVE View Deprecation

Oracle is replacing the view V$PQ_SLAVE with V$PX_SERVER in Oracle Database 23ai.

Date: April 2023

Use V$PX_SERVER in place of V$PQ_SLAVE.

V$FS_FAILOVER_STATS View Deprecation

The view V$FS_FAILOVER_STATS is deprecated and replaced by the view V$DG_BROKER_ROLE_CHANGE in Oracle Database 23ai.

Date: April 2023

The view V$FS_FAILOVER_STATS displayed one row with limited information about the last fast-start failover that occurred. The data was available in the primary database only, and was not

persisted across database restarts. The new view V$DG_BROKER_ROLE_CHANGE contains detailed information about the last ten role changes, including switchover, failover, and reinstate, across the Data Guard broker configuration, and provides richer information than the deprecated view V$FS_FAILOVER_STATS.

DBA_HANG_MANAGER_PARAMETERS Data Dictionary View Deprecation

The DBA_HANG_MANAGER_PARAMETERS static Data Dictionary view is deprecated with Oracle Database 23ai. Use DBA_BLOCKER_RESOLVER_PARAMETERS instead.

Date: July 2023

The DBA_HANG_MANAGER_PARAMETERS static Data Dictionary view shows the available user-tunable DBMS_BLOCKER_RESOLVER parameters and values, formerly available as DBA_HANG_MANAGER_PARAMETERS. This view is being replaced with DBA_BLOCKER_RESOLVER_PARAMETERS. DBA_HANG_MANAGER_PARAMETERS can be removed in a future release.

V$RECOVERY_SLAVE View Deprecation

Oracle is replacing the view V$RECOVERY_SLAVE with V$RECOVERY_WORKER in Oracle Database 23ai.

Date: August 2023

Use V$RECOVERY_WORKER in place of V$RECOVERY_SLAVE.

Deprecated Parameters in Oracle Database 23ai

As part of your upgrade plan, review the initialization parameters listed here that are deprecated starting with this Oracle Database release.

To obtain the current list of all deprecated parameters in your database release, run the following query in SQL*Plus:

```
SELECT name from v$parameter
         WHERE isdeprecated = 'TRUE' ORDER BY name;
```

- ENCRYPT_NEW_TABLESPACES Deprecation
  Starting with Oracle Database 23ai, the ENCRYPT_NEW_TABLESPACES initialization parameter is deprecated.
- ALLOW_MD5_CERTS and ALLOW_SHA1_CERTS sqlnet.ora Parameter Deprecation
  Starting in Oracle Database 23ai, the ALLOW_MD5_CERTS and ALLOW_SHA1_CERTS sqlnet.ora parameters are deprecated.
- MY_WALLET_DIRECTORY Connect String Deprecated
  The connect string parameter MY_WALLET_DIRECTORY has been deprecated with Oracle Database 23ai.
- FIPS Parameters Deprecated
  Starting with Oracle Database 23ai, several parameters associated with FIPS_140 are deprecated.
- ONE_STEP_PLUGIN_FOR_PDB_WITH_TDE Initialization Parameter Deprecated
  The parameter ONE_STEP_PLUGIN_FOR_PDB_WITH_TDE is deprecated with Oracle Database 23ai.

- Traditional Audit Initialization Parameters Deprecated
  With the desupport of creating and modifying traditional audit policies with Oracle Database 23, the traditional audit parameters are deprecated.
- PRE_PAGE_SGA Initialization Parameter Deprecated
- REMOTE_OS_ROLES Initialization Parameter Deprecated
  The `REMOTE_OS_ROLES` initialization parameter is deprecated in Oracle Database 23ai

ENCRYPT_NEW_TABLESPACES Deprecation

Starting with Oracle Database 23ai, the `ENCRYPT_NEW_TABLESPACES` initialization parameter is deprecated.

Date: April 2023

Oracle recommends that you use the initialization parameter `TABLESPACE_ENCRYPTION`, which is new for Oracle Database 23ai.

ALLOW_MD5_CERTS and ALLOW_SHA1_CERTS sqlnet.ora Parameter Deprecation

Starting in Oracle Database 23ai, the `ALLOW_MD5_CERTS` and `ALLOW_SHA1_CERTS` `sqlnet.ora` parameters are deprecated.

Date: April 2023

Instead of these parameters, use the `ALLOWED_WEAK_CERT_ALGORITHMS` `sqlnet.ora` parameter, which is new with Oracle Database 23ai.

MY_WALLET_DIRECTORY Connect String Deprecated

The connect string parameter `MY_WALLET_DIRECTORY` has been deprecated with Oracle Database 23ai.

Date: April 2023

Oracle recommends that you use `WALLET_LOCATION` in the connect string to override the `sqlnet.ora` `WALLET_LOCATION` setting. `WALLET_LOCATION` has been updated for connect strings so that the same parameter can be used in `sqlnet.ora` and in `tnsnames.ora` This change simplifies the parameters that you need to remember. Oracle recommends that you change your client connect strings to use `WALLET_LOCATION` instead of `MY_WALLET_DIRECTORY`.

FIPS Parameters Deprecated

Starting with Oracle Database 23ai, several parameters associated with `FIPS_140` are deprecated.

Date: April 2023

`FIPS_140` in `FIPS.ORA` can be used to enable `FIPS` for all features starting with Oracle Database 23ai. The following FIPS parameters are deprecated:

- SQLNET.ORA: `FIPS_140` to enable FIPS for native network encryption
- FIPS.ORA: `SSLFIPS_140` to enable FIPS for TLS
- Initialization parameter: `DBFIPS_140` to enable FIPS for TDE and `DBMS_CRYPTO`

ONE_STEP_PLUGIN_FOR_PDB_WITH_TDE Initialization Parameter Deprecated

The parameter ONE_STEP_PLUGIN_FOR_PDB_WITH_TDE is deprecated with Oracle Database 23ai.

Date: July 2023

ONE_STEP_PLUGIN_FOR_PDB_WITH_TDE enables you to clone remotely or to relocate encrypted PDBs without providing the Transparent Data Encryption (TDE) keystore password. However, EXTERNAL STORE provides the same functionality, and is universally applicable to all ADMINISTER KEY MANAGEMENT statements that do not change the TDE configuration. Instead of using ONE_STEP_PLUGIN_FOR_PDB_WITH_TDE, Oracle recommends that you use the IDENTIFIED BY EXTERNAL STORE clause for the ADMINISTER KEY MANAGEMENT statement.

Traditional Audit Initialization Parameters Deprecated

With the desupport of creating and modifying traditional audit policies with Oracle Database 23, the traditional audit parameters are deprecated.

Date: July 2023

Traditional audit policies are available after upgrading, but the initialization parameters associated with those policies are deprecated. These deprecated parameters include the following:

- AUDIT_TRAIL
- AUDIT_SYS_OPERATIONS
- AUDIT_FILE_DEST
- AUDIT_SYSLOG_LEVEL

Oracle recommends that you migrate to unified auditing as soon as possible, because these traditional audit parameters can be removed in a future database release.

PRE_PAGE_SGA Initialization Parameter Deprecated

Oracle is deprecating PRE_PAGE_SGA because it is obsolete. Setting the parameter typically provides little or no potential performance benefits, and can create problems.
Date: August 2023

The Oracle Database design for SGA packaging has evolved over time, so that process start-up effects on the SGA are initiated after instance startup, and there is little to no benefit in changing the value for PRE_PAGE_SGA. The only use case for this parameter is on Oracle Exadata systems, which should have PRE_PAGE_SGA set to TRUE.

REMOTE_OS_ROLES Initialization Parameter Deprecated

The REMOTE_OS_ROLES initialization parameter is deprecated in Oracle Database 23ai

Date: January 2025

The REMOTE_OS_ROLES instructed the database to grant database roles based on the user's operating system group information in the database client. This feature was particularly for use with the initialization parameter REMOTE_OS_AUTHENT, which was deprecated, and desupported with Oracle Database 21c. The Oracle Database client can no longer authenticate a user within Oracle Database. This initialization parameter accordingly no longer has a supported use case, and is now deprecated.

Oracle Database 21c Behavior Changes, Desupports, and Deprecations

Review for descriptions of Oracle Database 21c release changes.

- Behavior Changes for Oracle Database 21c Upgrade Planning
  Review these behavior changes to help plan for upgrades to Oracle Database 21c.
- Desupported Features in Oracle Database 21c
  As part of your upgrade plan, review the desupported features in this Oracle Database release.
- Desupported Views in Oracle Database 21c
  Review this list of desupported views for changes and replacements in view settings in this release.
- Desupported Initialization Parameters in Oracle Database 21c
  Review this list of desupported initialization parameters for changes and replacements in parameter settings in this release.
- Deprecated Features in Oracle Database 21c
  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 Views in Oracle Database 21c
  As part of your upgrade plan, review the views that are deprecated starting with this Oracle Database release.
- Deprecated Parameters in Oracle Database 21c
  As part of your upgrade plan, review the initialization parameters that are deprecated starting with this Oracle Database release.

Behavior Changes for Oracle Database 21c Upgrade Planning

Review these behavior changes to help plan for upgrades to Oracle Database 21c.

Behavior changes can include default changes for parameters or features, product name changes, and other changes to the database configuration that may require your attention.

- About Read-Only Oracle Homes in Oracle Database 21c
  Starting with Oracle Database 21c, an Oracle Database installation configures all Oracle Database homes in read-only mode by default.
- Multitenant Upgrades Only in Oracle Database 21c
  Starting with Oracle Database 21c, Oracle Database is only supported using the multitenant architecture.
- Logical Standby and New Data Types
  If you use logical standby (when not used as part of `DBMS_ROLLING`), then you can use only data types added before Oracle Database 12c Release 2 (12.2)
- Relocation of HR Sample Schema
  Starting with Oracle Database 21c, the HR sample schema no longer ships as part of Oracle Database.
- Manage DRCP on PDBs
  Starting with Oracle Database 21c, Database Resident Connection Pool (DRCP) can be configured and managed for individual PDBs.

- Oracle Advanced Cluster File System (Oracle ACFS) Name Change
  Starting with Oracle Database 21c, the name of Oracle Automatic Storage Management Cluster File System (Oracle ACFS) is changed to Oracle Advanced Cluster File System (Oracle ACFS).
- Windows Authentication No Longer Uses NTLM by Default
  For Microsoft Windows installations with `AUTHENTICATION_SERVICES=NTS`, in this Oracle Database release, the `SQLNET.NO_NTLM` parameter setting in the `sqlnet.ora` file defaults to `TRUE`, which can cause ORA-12638 errors.

About Read-Only Oracle Homes in Oracle Database 21c

Starting with Oracle Database 21c, an Oracle Database installation configures all Oracle Database homes in read-only mode by default.

> **Note:**
>
> This description of read-only Oracle home installation describes the default in Oracle Database 21c, where the only available configuration was a read-only Oracle home, and a software-only Oracle Database installation was not an option.

A read-only Oracle Home simplifies provisioning by implementing separation of installation and configuration.

Before Oracle Database 21c, the default `ORACLE_HOME` layout combined `ORACLE_HOME`, `ORACLE_BASE_HOME` and `ORACLE_BASE_CONFIG` into a single location. Starting with Oracle Database 21c, the only available configuration is a read-only `ORACLE_HOME` where `ORACLE_BASE_HOME` and `ORACLE_BASE_CONFIG` are located separately from `ORACLE_HOME`.

In a read-only Oracle home, all the configuration data and log files reside outside of the read-only Oracle home.

Apart from the traditional `ORACLE_BASE` and `ORACLE_HOME` directories, the following directories contain files that used to be in `ORACLE_HOME`:

- `ORACLE_BASE_HOME`
- `ORACLE_BASE_CONFIG`

> **Note:**
>
> This feature does not affect how database administrators monitor, diagnose, and tune their system performance.

Multitenant Upgrades Only in Oracle Database 21c

Starting with Oracle Database 21c, Oracle Database is only supported using the multitenant architecture.

> **Note:**
>
> A multitenant container database is the only supported architecture in Oracle Database 21c and later releases. While the documentation is being revised, legacy terminology may persist. In most cases, "database" and "non-CDB" refer to a CDB or PDB, depending on context. In some contexts, such as upgrades, "non-CDB" refers to a non-CDB from a previous release.

Logical Standby and New Data Types

If you use logical standby (when not used as part of `DBMS_ROLLING`), then you can use only data types added before Oracle Database 12c Release 2 (12.2)

Date: May 2021

New data types added after Oracle Database 12c Release 1 (12.1) are not supported with Oracle Data Guard logical standby. For example, Oracle Data Guard logical standby does not support long identifiers, complex Abstract Data Types (ADTs), and spatial data types. Note that this limitation does not exist with an Oracle Data Guard physical standby database, `DBMS_ROLLING`, or Oracle GoldenGate. To obtain the benefits of a standby database with more recent data types, Oracle recommends that you consider using either a physical standby database, a snapshot standby database, or that you use the logical replication features of Oracle GoldenGate.

Relocation of HR Sample Schema

Starting with Oracle Database 21c, the HR sample schema no longer ships as part of Oracle Database.

To install the HR schema, refer to the section "Installing Sample Schemas" in *Oracle Database Database Sample Schemas*

Related Topics

- Installation of Sample Schemas

Manage DRCP on PDBs

Starting with Oracle Database 21c, Database Resident Connection Pool (DRCP) can be configured and managed for individual PDBs.

When you choose to configure DRCP so that it is managed from individual PDBs, you can configure, manage, and monitor pools on individual pluggable databases (PDBs).

In previous releases, the DRCP pool was used by the entire container database (CDB). This feature eases the management of DRCP pool by changing the granularity of the DRCP pool from the entire CDB to a DRCP pool for individual PDBs. This change enables tenant administrators to configure and manage independent tenant-specific DRCP pools.

This feature is enabled by a new database parameter, `ENABLE_PER_PDB_DRCP`. By default the parameter is set to `ENABLE_PER_PDB_DRCP=FALSE`. If you set `ENABLE_PER_PDB_DRCP=TRUE`, then this feature is enabled.

When you upgrade a PDB to Oracle Database 21c, the default connection pool for the PDB is created in the PDB. If you then enable this feature with the initialization parameter, then

instead of managing connection pooling at the CDB, the connection pool is managed locally by the PDB.

Oracle Advanced Cluster File System (Oracle ACFS) Name Change

Starting with Oracle Database 21c, the name of Oracle Automatic Storage Management Cluster File System (Oracle ACFS) is changed to Oracle Advanced Cluster File System (Oracle ACFS).

This change is only a change of the name. The basic function of Oracle's cluster file system continues to be the same. Oracle continues to develop and enhance Oracle ACFS.

Windows Authentication No Longer Uses NTLM by Default

For Microsoft Windows installations with `AUTHENTICATION_SERVICES=NTS`, in this Oracle Database release, the `SQLNET.NO_NTLM` parameter setting in the `sqlnet.ora` file defaults to `TRUE`, which can cause ORA-12638 errors.

In previous releases, the default for `SQLNET.NO_NTLM` was `FALSE`. `SQLNET.NO_NTLM` controls whether `NTLM` can be used with NTS authentication (`AUTHENTICATION_SERVICES=NTS`). A `TRUE` setting means that NTLM cannot be used in NTS authentication. Because NTLM does not normally provide mutual authentication and is hence less secure, a `TRUE` setting for `SQLNET.NO_NTLM` makes the database and client more secure.

The `SQLNET.NO_NTLM` parameter is used on both the server and the client. If you have upgraded a Microsoft Windows installation of Oracle Database, or upgraded a client in which `SQLNET.NO_NTLM` had not been set, then its default will be `TRUE`. In that case, when you have `SQLNET.AUTHENTICATION_SERVICES=NTS` in your `sqlnet.ora`, clients can encounter the error `ORA-12638: Credential retrieval failed`.

If you prefer to use NTLM authentication for certain clients, then set this parameter as required in client-side `sqlnet.ora` files:

```
SQLNET.NO_NTLM=FALSE
```

You must include this setting on both the server and client, and this setting should be the same on both. Ideally, you should ensure that `SQLNET.NO_NTLM` is set to `TRUE`. However, if there is an authentication failure in `extproc`, a virtual account, or a local account on Windows, set the client `SQLNET.NO_NTLM` to `FALSE`, and then retry the login. If you change `SQLNET.NO_NTLM` on the server, then you must restart the database.

Related Topics

- Guideline for Securing Authentication for Oracle Database Microsoft Windows Installations
- Upgrade Your Database Now - ORA-12638 on Windows only from Oracle 19.10.0 onwards
- Windows : While Connect to Database Getting ORA-12638 After Applying Jan 2021 WINDBBP 19.10.0.0.210119 On Windows 64/32 Bit Database Client (Doc ID 2757734.1)

Desupported Features in Oracle Database 21c

As part of your upgrade plan, review the desupported features in this Oracle Database release.

- Desupport of DBMS_OBFUSCATION_TOOLKIT Package
  Starting in Oracle Database 21c, the package `DBMS_OBFUSCATION_TOOLKIT` is desupported, and replaced with `DBMS_CRYPTO`.

- Desupport of Several XML Database (XDB) features
  Starting with Oracle Database 21c, several XML Database features are desupported.
- Desupport of DBMS_LOB.LOADFROMFILE and LOB Buffering
  Starting in Oracle Database 21c, the Large Object (LOB) features `DBMS_LOB.LOADFROMFILE` and LOB buffering are desupported.
- Desupport of Oracle Data Guard Broker Properties and Logical Standby
  Oracle Data Guard broker properties and logical standby properties whose functionality is replaced with the new EDIT … SET PARAMETER command in DGMGRL are now desupported.
- Desupport of DBMS_CRYPTO_TOOLKIT_TYPES and DBMS_CRYPTO_TOOLKIT
  Starting with Oracle Database 21c, the data types `DBMS_CRYPTO_TOOLKIT_TYPES` and package `DBMS_CRYPTO_TOOLKIT` are desupported.
- Desupport of Non-CDB Oracle Databases
  Starting with Oracle Database 21c, installation of non-CDB Oracle Database architecture is no longer supported.
- Desupport of Cluster Domain Member Clusters
  Effective with Oracle Grid Infrastructure 21c, Member Clusters, which are part of the Oracle Cluster Domain architecture, are desupported.
- Desupport of Unicode Collation Algorithm (UCA) 6.1 Collations
  Starting with Oracle Database 21c, the Unicode Collation Algorithm (UCA) 6.1 collations (`UCA0610_*`) are desupported. Use UCA 12.1 instead.
- Desupport of ACFS on Microsoft Windows
  Starting with Oracle Database 21c, the Oracle Grid Infrastructure feature Oracle Advanced Cluster File System (Oracle ACFS) is desupported with Microsoft Windows
- Desupport of Oracle ACFS Security (Vault) and ACFS Auditing
  Starting with Oracle Grid Infrastructure 21c, Oracle Advanced Cluster File System (ACFS) Security (Vault) and ACFS Auditing are desupported.
- Desupport of Oracle ACFS on Member Clusters (ACFS Remote)
  Starting with Oracle Grid Infrastructure 21c, Oracle Advanced Cluster File System (ACFS) on Member Clusters (ACFS Remote) is desupported.
- Desupport of ACFS Encryption on Solaris and Windows
  Starting with Oracle Database 21c, Oracle ACFS Encryption is desupported with no replacement on Oracle Solaris and Microsoft Windows.
- Desupport of ACFS Replication REPV1
  Starting with Oracle Database 21c, the Oracle ACFS replication protocol `repv1` is desupported.
- Desupport of Vendor Clusterware Integration with Oracle Clusterware
  Starting with Oracle Clusterware 21c, the integration of vendor or third party clusterware with Oracle Clusterware is desupported.
- Desupport of VERIFY_FUNCTION and VERIFY_FUNCTION_11G
  The `VERIFY_FUNCTION` and `VERIFY_FUNCTION_11G` password verify functions are desupported in Oracle Database 21c.
- Desupport of Deprecated Oracle Database Vault Roles
  The Oracle Database Vault roles `DV_PUBLIC`, `DV_REALM_OWNER`, and `DV_REALM_RESOURCE` are desupported in Oracle Database 21c.
- Desupport of Anonymous RC4 Cipher Suite
  The use of the anonymous RC4 cipher suite for non-authenticated TLS connections is desupported in Oracle Database 21c (`SSL_DH_anon_WITH_RC4_128_MD5`)

- Desupport of Adobe Flash-Based Oracle Enterprise Manager Express
  Flash-based Oracle EM Express is desupported in Oracle 21c. Use the JET-based Oracle EM Express, which is the default.
- Desupport of Intelligent Data Placement (IDP)
  Intelligent Data Placement (IDP) is desupported in Oracle Database 21c.
- Desupport of XML DB Content Connector
  Oracle XML DB Content Connector was deprecated in Oracle Database 12c Release 2. It is desupported and removed in Oracle Database 21c.
- Desupport of DBMS_XMLSAVE
  The PL/SQL package `DBMS_XMLSAVE` is desupported in Oracle Database 21c. Use `DBMS_XMLSTORE` instead.
- Desupport of DBMS_XMLQUERY
  The PL/SQL package `DBMS_XMLQUERY` is desupported in Oracle Database 21c. Use `DBMS_XMLGEN` instead.
- Desupport of FIPS Protect and Process Strength 0
  The protect and process strength 0 (RSA key length 512) equivalent is desupported for FIPS. This strength is still available for the non-FIPS mode.
- Desupport of PDB Flat File Dictionary Dumps
  The ability to create flat file dictionary dumps of pluggable databases (PDBs) is desupported in Oracle Database 21c.
- Desupport of Oracle Fail Safe
  Starting with Oracle Database 21c, Oracle Fail Safe is desupported for Oracle Database releases.
- Desupport of EUS Current User Database Links
  Current user database links for Enterprise User Security (EUS) are not supported in Oracle Database 21c.

Desupport of DBMS_OBFUSCATION_TOOLKIT Package

Starting in Oracle Database 21c, the package `DBMS_OBFUSCATION_TOOLKIT` is desupported, and replaced with `DBMS_CRYPTO`.

Date: May 2021

`DBMS_OBFUSCATION_TOOLKIT` was deprecated in Oracle Database 10g Release 2. It is now removed in Oracle Database 21c. `DBMS_CRYPTO` replaces the functionality that `DBMS_OBFUSCATION_TOOLKIT` provided previously. `DBMS_CRYPTO` includes more modern and secure encryption technologies for your security requirements.

Desupport of Several XML Database (XDB) features

Starting with Oracle Database 21c, several XML Database features are desupported.

Date: May 2021

The following features are desupported:

- Package `DBMS_XDBT`. There is no replacement.
- 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.
- Oracle XML DB Content Connector.

Desupport of DBMS_LOB.LOADFROMFILE and LOB Buffering

Starting in Oracle Database 21c, the Large Object (LOB) features `DBMS_LOB.LOADFROMFILE` and LOB buffering are desupported.

Date: May 2021

The following features are desupported:

- `DBMS_LOB.LOADFROMFILE` Procedure. Use `DBMS_LOB.LoadClobFromFile` or `DBMS_LOB.LoadBlobFromFile` instead.
- The LOB buffering subsystem APIs:
  - `OCILobEnableBuffering()`
  - `OCILobDisableBuffering()`
  - `OCILobFlushBuffer()`

  In place of using these LOB buffering functions, use the LOB prefetch feature.

Desupport of Oracle Data Guard Broker Properties and Logical Standby

Oracle Data Guard broker properties and logical standby properties whose functionality is replaced with the new EDIT … SET PARAMETER command in DGMGRL are now desupported.

Date: May 2021

The following Oracle Data Guard broker properties are desupported in Oracle Database 21c:

- `ArchiveLagTarget`
- `DataGuardSyncLatency`
- `LogArchiveMaxProcesses`
- `LogArchiveMinSucceedDest`
- `LogArchiveTrace`
- `StandbyFileManagement`
- `DbFileNameConvert`
- `LogArchiveFormat`
- `LogFileNameConvert`

The following Oracle Data Guard broker Properties that affect Logical Standby are desupported in Oracle Database 21c:

- LsbyMaxEventsRecorded
- LsbyMaxServers
- LsbyMaxSga
- LsbyPreserveCommitOrder
- LsbyRecordAppliedDdl
- LsbyRecordSkippedDdl
- LsbyRecordSkipErrors
- LsbyParameter

Desupport of DBMS_CRYPTO_TOOLKIT_TYPES and DBMS_CRYPTO_TOOLKIT

Starting with Oracle Database 21c, the data types DBMS_CRYPTO_TOOLKIT_TYPES and package DBMS_CRYPTO_TOOLKIT are desupported.

Date: May 2021

The data types DBMS_CRYPTO_TOOLKIT_TYPES and the DBMS_CRYPTO_TOOLKIT package were deprecated in Oracle9i Database. These data types and package are now removed from Oracle Database 21c.

Desupport of Non-CDB Oracle Databases

Starting with Oracle Database 21c, installation of non-CDB Oracle Database architecture is no longer supported.

Date: May 2021

The non-CDB architecture was deprecated in Oracle Database 12c. It is desupported in Oracle Database 21c. Oracle Universal Installer can no longer be used to create non-CDB Oracle Database instances.

Desupport of Cluster Domain Member Clusters

Effective with Oracle Grid Infrastructure 21c, Member Clusters, which are part of the Oracle Cluster Domain architecture, are desupported.

Date: May 2021

Desupporting 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. It also includes Remote Oracle Advanced Cluster File System (Remote ACFS). Member Clusters were first introduced to simplify the management of larger cluster estates, and to minimize outage times for certain failures and configurations. However, additional enhancements in Standalone Clusters provide the same benefits. These enhancements make the use of Member Clusters unnecessary. Consequently, if you are currently using Member Clusters, then Oracle recommends that you use Standalone Clusters going forward.

Desupport of Unicode Collation Algorithm (UCA) 6.1 Collations

Starting with Oracle Database 21c, the Unicode Collation Algorithm (UCA) 6.1 collations (UCA0610_*) are desupported. Use UCA 12.1 instead.

Date: May 2021

Oracle recommends that you use the latest supported version of Unicode Collation Algorithm (UCA) collations, which in Oracle Database 21c is UCA 12.1. UCA 6.1 collations were deprecated in Oracle Database 12c Release 2. UCA 12.1 incorporates all of the UCA enhancements since version 6.1, as well as proper collation weight assignments for all new characters introduced since Unicode 6.1.

Desupport of ACFS on Microsoft Windows

Starting with Oracle Database 21c, the Oracle Grid Infrastructure feature Oracle Advanced Cluster File System (Oracle ACFS) is desupported with Microsoft Windows

Date: May 2021

Oracle ACFS is used for two major use cases:

- Oracle Database Files for Oracle Real Application Clusters (Oracle RAC)
- Generic files (unstructured data) that need to be shared across multiple hosts.

For Oracle Real Application Clusters files, Oracle recommends that you use Oracle ASM. For generic files, depending on your use case, Oracle recommends that you either move files to Oracle Database File System (DBFS), or move files to Microsoft Windows shared files.

Desupport of Oracle ACFS Security (Vault) and ACFS Auditing

Starting with Oracle Grid Infrastructure 21c, Oracle Advanced Cluster File System (ACFS) Security (Vault) and ACFS Auditing are desupported.

Date: May 2021

To manage security and auditing, Oracle recommends that you use your operating system access controls and auditing systems. For example, with Linux, you can use the Linux Auditing System. As part of this desupport, the following views are removed:

```
V$ASM_ACFS_SEC_ADMIN
V$ASM_ACFS_SEC_CMDRULE
V$ASM_ACFS_SEC_REALM
V$ASM_ACFS_SEC_REALM_FILTER
V$ASM_ACFS_SEC_REALM_GROUP
V$ASM_ACFS_SEC_REALM_USER
V$ASM_ACFS_SEC_RULE
V$ASM_ACFS_SEC_RULESET
V$ASM_ACFS_SEC_RULESET_RULE
V$ASM_ACFS_SECURITY_INFO
```

Desupport of Oracle ACFS on Member Clusters (ACFS Remote)

Starting with Oracle Grid Infrastructure 21c, Oracle Advanced Cluster File System (ACFS) on Member Clusters (ACFS Remote) is desupported.

Date: May 2021

Oracle Advanced Cluster File System (ACFS) on Member Clusters (ACFS Remote) is desupported. Desupporting certain clustering features with limited adoption allows Oracle to focus on improving core scaling, availability, and manageability across all features and functionality.

Desupport of ACFS Encryption on Solaris and Windows

Starting with Oracle Database 21c, Oracle ACFS Encryption is desupported with no replacement on Oracle Solaris and Microsoft Windows.

Date: May 2021

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.

Desupport of ACFS Replication REPV1

Starting with Oracle Database 21c, the Oracle ACFS replication protocol `repv1` is desupported.

Date: May 2021

The initial ACFS replication protocol `repv1` was released with Oracle Database 11g Release 2 (11.2).The `replv2` protocol has been required since Oracle Database 12c Release 2 (12.2). The `replv1` protocol was available only during the required upgrade to `replv2` in 12.2 and later releases. The `replv1` protocol was deprecated in Oracle Database 19c. It is desupported in Oracle Database 21c.

Desupport of Vendor Clusterware Integration with Oracle Clusterware

Starting with Oracle Clusterware 21c, the integration of vendor or third party clusterware with Oracle Clusterware is desupported.

Date: May 2021

The integration of vendor clusterware with Oracle Clusterware is desupported in Oracle Database 21c. Desupporting 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. This desupport applies to clustered deployments, including Oracle Real Application Clusters and Oracle RAC One Node.

Related Topics

- About vendor clusterware support for Oracle Clusterware (Doc ID 3044222.1)

Desupport of VERIFY_FUNCTION and VERIFY_FUNCTION_11G

The `VERIFY_FUNCTION` and `VERIFY_FUNCTION_11G` password verify functions are desupported in Oracle Database 21c.

Date: May 2021

These older functions are desupported because they enforce the weaker password restrictions from earlier releases. Instead, use the functions `ORA12C_VERIFY_FUNCTION`, `ORA12C_STRONG_VERIFY_FUNCTION`, or `ORA12C_STIG_VERIFY_FUNCTIONS`. These functions enforce stronger, more up-to-date password verification restrictions.

Desupport of Deprecated Oracle Database Vault Roles

The Oracle Database Vault roles `DV_PUBLIC`, `DV_REALM_OWNER`, and `DV_REALM_RESOURCE` are desupported in Oracle Database 21c.

Date: May 2021

Oracle deprecated these Oracle Database Vault roles in Oracle Database 19c. The roles are granted powerful privileges,but were seldom used. During upgrades to Oracle Database 21c, these roles are removed, and are not available in new Oracle Database installations. If you are using these roles, and you are upgrading your database, then note the roles and privileges granted to them and then create roles for these privileges after the upgrade is complete.

Desupport of Anonymous RC4 Cipher Suite

The use of the anonymous RC4 cipher suite for non-authenticated TLS connections is desupported in Oracle Database 21c (`SSL_DH_anon_WITH_RC4_128_MD5`)

Date: May 2021

Oracle recommends that you use the more secure authenticated connections available with Oracle Database. If you use anonymous Diffie-Hellman with RC4 for connecting to Oracle Internet Directory for Enterprise User Security, then you must migrate to use a different algorithm connection. Oracle recommends that you use either TLS one-way, or mutual authentication using certificates.

Desupport of Adobe Flash-Based Oracle Enterprise Manager Express

Flash-based Oracle EM Express is desupported in Oracle 21c. Use the JET-based Oracle EM Express, which is the default.

Date: May 2021

Adobe Flash is desupported by all major browsers starting in January 2021. Oracle Database 19c and later releases of Oracle Enterprise Manager Express (Oracle EM Express) use Oracle JavaScript Extension Toolkit (JET) technology as a replacement for Flash.

Desupport of Intelligent Data Placement (IDP)

Intelligent Data Placement (IDP) is desupported in Oracle Database 21c.

Date: May 2021

Intelligent Data Placement helped to place data on physical storage disks to reduce latency. This feature was deprecated in Oracle Database 12c release 2 (12.2). There is no replacement. Views and ASM disk group attributes associated with this feature are also desupported.

Desupport of XML DB Content Connector

Oracle XML DB Content Connector was deprecated in Oracle Database 12c Release 2. It is desupported and removed in Oracle Database 21c.

Date: May 2021

XML DB Content Connector implemented the Java standard JSR-170 API. This standard has been replaced by Java standard JSR-283.

Desupport of DBMS_XMLSAVE

The PL/SQL package `DBMS_XMLSAVE` is desupported in Oracle Database 21c. Use `DBMS_XMLSTORE` instead.

Date: May 2021

Replace `DBMS_XMLSAVE` calls with `DBMS_XMLSTORE`.

Desupport of DBMS_XMLQUERY

The PL/SQL package `DBMS_XMLQUERY` is desupported in Oracle Database 21c. Use `DBMS_XMLGEN` instead.

Date: May 2021

Replace calls to `DBMS_XMLQUERY` with `DBMS_XMLGEN`.

Desupport of FIPS Protect and Process Strength 0

The protect and process strength 0 (RSA key length 512) equivalent is desupported for FIPS. This strength is still available for the non-FIPS mode.

Date: May 2021

The default protect and process strength for Federal Information Processing Standard (FIPS) mode is currently 80. The equivalent RSA key strength is 1024. The equivalent ECC key strength is curves with minimum ECC curve key length 160, ECC named curves P192, K163 and B163 and above. The equivalent DH/DSA (Diffie Hellman, Digital Signature Algorithm) key length is 1024. Oracle recommends that you use FIPS protect strength 112, because the default FIPS protect strength, 80, is deprecated.

Desupport of PDB Flat File Dictionary Dumps

The ability to create flat file dictionary dumps of pluggable databases (PDBs) is desupported in Oracle Database 21c.

Date: May 2021

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. Starting with Oracle Database 21c, 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.

Desupport of Oracle Fail Safe

Starting with Oracle Database 21c, Oracle Fail Safe is desupported for Oracle Database releases.

Date: June 2021

This desupport notice does not apply to earlier releases of Oracle Database. Oracle Fail Safe will continue to be supported for the lifetime of Oracle Database 19c. For Oracle Database releases starting with Oracle Database 21c, if you are running Oracle Database on Microsoft

Windows, then Oracle recommends that you investigate other failover solutions, such as Oracle RAC One Node, or Oracle Database Standard Edition High Availability.

Desupport of EUS Current User Database Links

Current user database links for Enterprise User Security (EUS) are not supported in Oracle Database 21c.

Date: April 2024

Because Oracle Database releases after Oracle Database 19c only use the multitenant architecture, the use of EUS current user database links (`CREATE DATABASE LINK…TO CURRENT_USER`) is not supported in releases after Oracle Database 19c. Oracle recommends that you use fixed user or connected user database links with EUS.

Desupported Views in Oracle Database 21c

Review this list of desupported views for changes and replacements in view settings in this release.

- Desupport of V$OBJECT_USAGE View
  The view V$OBJECT_USAGE is desupported in Oracle Database 21c. Use USER_OBJECT_USAGE instead.

Desupport of V$OBJECT_USAGE View

The view V$OBJECT_USAGE is desupported in Oracle Database 21c. Use USER_OBJECT_USAGE instead.

Date: October 2021

The V$OBJECT_USAGE view displayed statistics about index usage gathered from the database for the indexes owned by the current user. This view was deprecated in Oracle Database 12c Release 1. It is now desupported. To obtain statistics about index usage owned by a user, use the USER_OBJECT_USAGE view instead of the V$OBJECT_USAGE view.

Desupported Initialization Parameters in Oracle Database 21c

Review this list of desupported initialization parameters for changes and replacements in parameter settings in this release.

- Desupport of UNIFIED_AUDIT_SGA_QUEUE_SIZE
  Starting in Oracle Database 21c, the initialization parameter `UNIFIED_AUDIT_SGA_QUEUE_SIZE` is desupported.
- Desupport of IGNORECASE Parameter for Passwords
  Starting in Oracle Database 21c, the `IGNORECASE` parameter for the `orapwd` file is desupported. All newly created password files are case-sensitive.
- Desupport of DISABLE_DIRECTORY_LINK_CHECK
  Starting in Oracle Database 21c, the `DISABLE_DIRECTORY_LINK_CHECK` parameter is desupported, with no replacement.
- Desupport of REMOTE_OS_AUTHENT Parameter
  The Oracle Database initialization parameter `REMOTE_OS_AUTHENT` has been removed from Oracle Database 21c.

- Desupport of SEC_CASE_SENSITIVE_LOGON
  The `SEC_CASE_SENSITIVE_LOGON` parameter is desupported in Oracle Database 21c.
- Desupport of CLUSTER_DATABASE_INSTANCES Parameter
  The `CLUSTER_DATABASE_INSTANCES` parameter is desupported in 21c. There is no replacement.

Desupport of UNIFIED_AUDIT_SGA_QUEUE_SIZE

Starting in Oracle Database 21c, the initialization parameter `UNIFIED_AUDIT_SGA_QUEUE_SIZE` is desupported.

May 2021

The `UNIFIED_AUDIT_SGA_QUEUE_SIZE` parameter was deprecated in Oracle Database 12c Release 2 (12.2), and the value for the parameter was no longer was honored. It is now removed.

Desupport of IGNORECASE Parameter for Passwords

Starting in Oracle Database 21c, the `IGNORECASE` parameter for the `orapwd` file is desupported. All newly created password files are case-sensitive.

May 2021

Case-sensitive password files provide more security than older non-case sensitive password files. To enhance security, Oracle recommends that you use case-sensitive passwords. However, upgraded password files from earlier Oracle Database releases can retain original case-insensitive passwords. To ensure that password files are case-sensitive, Oracle recommends that you force case sensitivity by migrating password files from one format to another, using the following syntax: `orapwd input_file=`*`input_password _file`* `file=`*`output_password_file`*

Desupport of DISABLE_DIRECTORY_LINK_CHECK

Starting in Oracle Database 21c, the `DISABLE_DIRECTORY_LINK_CHECK` parameter is desupported, with no replacement.

May 2021

The `DISABLE_DIRECTORY_LINK_CHECK` parameter is disabled. Symbolic links managed previously with this parameter fail in the new Oracle Database release. 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`.

Desupport of REMOTE_OS_AUTHENT Parameter

The Oracle Database initialization parameter `REMOTE_OS_AUTHENT` has been removed from Oracle Database 21c.

May 2021

`REMOTE_OS_AUTHENT` was deprecated in Oracle Database 11g. To prevent potentially insecure connections to the database, Oracle is removing this authentication option in Oracle Database 21c.

Desupport of SEC_CASE_SENSITIVE_LOGON

The `SEC_CASE_SENSITIVE_LOGON` parameter is desupported in Oracle Database 21c.

May 2021

The SEC_CASE_SENSITIVE_LOGON parameter was deprecated in Oracle Database 12c Release 1 (12.1). To ensure that new passwords are case-sensitive, Oracle is removing this parameter from Oracle Database 21c.

Related Topics

- Checking for Accounts Using Case-Insensitive Password Version

Desupport of CLUSTER_DATABASE_INSTANCES Parameter

The `CLUSTER_DATABASE_INSTANCES` parameter is desupported in 21c. There is no replacement.

October 2021

The `init.ora` parameter `CLUSTER_DATABASE_INSTANCES` specified the number of configured Oracle Real Application Clusters (Oracle RAC) instances. This parameter was deprecated in Oracle Database 19c, because 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.

Deprecated Features in Oracle Database 21c

As part of your upgrade plan, review the features that are deprecated in this Oracle Database release, and review alternatives for your application strategies.

- Deprecation of AUTO OPTIMIZE Framework
  In Oracle Database Release 21c, the procedures `ADD_AUTO_OPTIMIZE` and `REMOVE_AUTO_OPTIMIZE`, and the views `CTX_AUTO_OPTIMIZE_INDEXES`, `CTX_USER_AUTO_OPTIMIZE_INDEXES` and `CTX_AUTO_OPTIMIZE_STATUS` are deprecated.
- Deprecation of CTXFILTERCACHE Query Operator
  Starting in Oracle Database Release 21c, `CTXFILTERCACHE` is deprecated, and also `CTX_FILTER_CACHE_STATISTICS` and `QUERY_FILTER_CACHE_SIZE`.
- Deprecation of Policy-Managed Databases
  Starting with Oracle Grid Infrastructure 21c, policy-managed databases are deprecated.
- Deprecation of Traditional Auditing
  Traditional auditing is deprecated in Oracle Database 21c. Oracle recommends that you use unified auditing, which enables selective and more effective auditing inside Oracle Database.
- Deprecation of Older Encryption Algorithms
  Starting with Oracle Database 21c, older encryption and hashing algorithms are deprecated.
- Deprecation of Cluster Domain - Domain Services Cluster
  Starting with Oracle Grid Infrastructure 21c, Domain Services Clusters (DSC), which is part of the Oracle Cluster Domain architecture, are deprecated.

- Deprecation of Enterprise User Security (EUS) User Migration Utility
  Enterprise User Security (EUS) User Migration Utility (UMU) is deprecated in Oracle Database 21c. Use EUS Manager (EUSM) features instead.
- Logical Standby and New Data Types
  If you use logical standby (when not used as part of `DBMS_ROLLING`), then you can use only data types added before Oracle Database 12c Release 2 (12.2)
- Deprecation of Sharded Queues
  AQ sharded queues are deprecated in Oracle Database 21c. Use Transactional Event Queues (TEQ) instead.
- Deprecation of MySQL Client Library Driver for Oracle
  The MySQL Client Library Driver for Oracle is deprecated in Oracle Database 21c.
- Deprecation of TLS 1.0 and 1.1 Transport Layer Security
  Starting with Oracle Database 21c, Transport Layer Security protocol versions 1.0 and 1.1 (TLS 1.0 and TLS 1.1) are deprecated.
- Deprecation of Unix Crypt (or MD5crypt) Password Verifier
  The Unix Crypt (`MD5crypt`) password verifier algorithm is deprecated in Oracle Database 21c server and clients, and passwords using this algorithm will stop working in a future release.
- Deprecation of ODP.NET OracleConfiguration.DirectoryType Property
  The Oracle Data Provider for .NET `OracleConfiguration DirectoryType` property and .NET configuration file `DIRECTORY_TYPE` setting are deprecated in Oracle Database 21c, and can be desupported in a future release.
- Deprecation of Weaker Encryption Key Strengths
  The use of weaker encryption keys is deprecated in Oracle Database 21c.
- Deprecation of DBSNMP Packages for Adaptive Thresholds Feature
  DBSNMP PL/SQL packages associated with the Adaptive Thresholds feature are deprecated in Oracle Database 21c.
- Deprecation of Oracle GoldenGate Replication for Oracle Sharding High Availability
  Oracle GoldenGate replication support for Oracle Sharding High Availability is deprecated in Oracle Database 21c.
- Deprecation of Anonymous Cipher Suites with Outbound TLS Connections
  Anonymous cipher suites for outbound TLS connections (Database to other services) are deprecated with Oracle Database 21c.
- Deprecation of the KERBEROS5PRE Adapter
  The use of the `KERBEROS5PRE` adapter is deprecated with Oracle Database 21c. Oracle recommends that you use the `KERBEROS5` adapter instead.
- Deprecation of Oracle Wallet Manager
  Oracle Wallet Manager (OWM) is deprecated with Oracle Database 21c.
- Deprecation of Oracle Enterprise Manager Database Express
  Oracle Enterprise Manager Database Express (EM Express) is deprecated, and will be removed in a future Oracle Database release.
- Deprecation of Service Attribute Value SESSION_STATE_CONSISTENCY = STATIC
  The combination of session attribute values `FAILOVER_TYPE = TRANSACTION` with `SESSION_STATE_CONSISTENCY = STATIC` is deprecated in Oracle Database 21c.
- Deprecation of SHA-1 use for SQLNET and DBMS_CRYPTO
  The use of SHA-1 with `DBMS_CRYPTO`, `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` and `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` is deprecated.

- Deprecation of ACFSUTIL REPL REVERSE
  The `acfsutil repl reverse` command is deprecated in Oracle Database 21c. Use `repl failover` or `repl switchover` instead.
- Deprecation of Oracle OLAP
  Analytic workspaces, the OLAP DML programming language, and the OLAP Java API are deprecated in Oracle Database 21c.
- Deprecation of Oracle Database Extensions for .NET
  Oracle Database Extensions for .NET is deprecated in Oracle Database 21c. Oracle recommends that you either place .NET code in the middle tier, or use the External Procedures feature, or rewrite the code using PL/SQL or Java.
- Deprecation of Repository Events
  The use of repository events to trigger application actions is deprecated in Oracle Database 21c (21.3). There is no replacement.
- Deprecation of Quality of Service Management
  Starting with Oracle Database Release 21c, Oracle Quality of Service Management (QoSM, or QOS Management) is deprecated.
- Grid Infrastructure Management Repository Deprecation
  The Grid Infrastructure Management Repository (GIMR), or Management Database (`MGMTDB`), is deprecated in Oracle Database 21c.

Deprecation of AUTO OPTIMIZE Framework

In Oracle Database Release 21c, the procedures `ADD_AUTO_OPTIMIZE` and `REMOVE_AUTO_OPTIMIZE`, and the views `CTX_AUTO_OPTIMIZE_INDEXES`, `CTX_USER_AUTO_OPTIMIZE_INDEXES` and `CTX_AUTO_OPTIMIZE_STATUS` are deprecated.

Date: May 2021

Basic optimization is now automatic for all indexes. You can also schedule additional optimization declaratively in the `CREATE INDEX` statement. Because of this enhancement, the `AUTO_OPTIMIZE` framework (procedures and views) is no longer necessary. Two procedures are deprecated in the `CTX_DDL` package: `ADD_AUTO_OPTIMIZE`, and `REMOVE_AUTO_OPTIMIZE`. In addition, the following views are deprecated: `CTX_AUTO_OPTIMIZE_INDEXES`, `CTX_USER_AUTO_OPTIMIZE_INDEXES` and `CTX_AUTO_OPTIMIZE_STATUS`.

Deprecation of CTXFILTERCACHE Query Operator

Starting in Oracle Database Release 21c, `CTXFILTERCACHE` is deprecated, and also `CTX_FILTER_CACHE_STATISTICS` and `QUERY_FILTER_CACHE_SIZE`.

Date: May 2021

The `CTXFILTERCACHE` query operator was designed to speed up commonly-used expressions in queries. In Oracle Database Release 21c, this function is replaced by other internal improvements. The `CTXFILTERCACHE` operator is deprecated (and will pass through its operands to be run as a normal query). Because they no longer have a function, the view `CTX_FILTER_CACHE_STATISTICS` is also deprecated, and also the storage attribute `QUERY_FILTER_CACHE_SIZE`.

Deprecation of Policy-Managed Databases

Starting with Oracle Grid Infrastructure 21c, policy-managed databases are deprecated.

Date: May 2021

You can continue to use existing server pools, and create new pools and policies. Resources using existing server pools can continue to use them transparently.

The use of CRS configuration policies and the CRS policy set can be desupported in a future release. In place of server pools and policy-managed databases, Oracle recommends that you use the new "Merged" management style.

Deprecation of Traditional Auditing

Traditional auditing is deprecated in Oracle Database 21c. Oracle recommends that you use unified auditing, which enables selective and more effective auditing inside Oracle Database.

Date: May 2021

Standard traditional auditing in Oracle Database has been provided for more than two decades. Traditional auditing provided built-in support for auditing statements, privileges and objects. Over the years, as data auditing became a key factor to ensuring the success of data strategy, Oracle recognized that there was a need to provide selective and effective auditing inside Oracle Database. To address this need, Oracle introduced unified auditing with Oracle Database 12c. In addition to providing built-in audit operation support, Unified Auditing simplifies management of auditing within the database, provides the ability to accelerate auditing based on conditions, and increases the security of audit data generated by the database. Unified Auditing and Traditional Auditing (mixed mode) has been the default auditing mode from Oracle Database 12c onward. Mixed mode auditing was offered as to enable you to become familiar with Unified Auditing, and to transition from Traditional Auditing. With the deprecation of Traditional Auditing in this release, Oracle recommends that you migrate to Unified Auditing. Refer to the migration procedure in Oracle Database Security Guide.

Deprecation of Older Encryption Algorithms

Starting with Oracle Database 21c, older encryption and hashing algorithms are deprecated.

Date: May 2021

The deprecated algorithms for `DBMS_CRYPTO` and native network encryption include MD4, MD5, DES, 3DES, and RC4-related algorithms as well as 3DES for Transparent Data Encryption (TDE). Removing older, less secure cryptography algorithms prevents accidental use of these algorithms. To meet your security requirements, Oracle recommends that you use more modern cryptography algorithms, such as the Advanced Encryption Standard (AES).

As a consequence of this deprecation, Oracle recommends that you review your network encryption configuration to see if you have specified use of any of the deprecated algorithms. If any are found, then switch to using a more modern cipher, such as AES. Also, if you are currently using 3DES encryption for your TDE deployment, then you should plan to migrate to a more modern algorithm such as AES. For more information, refer to *Oracle Database Security Guide*

Deprecation of Cluster Domain - Domain Services Cluster

Starting with Oracle Grid Infrastructure 21c, Domain Services Clusters (DSC), which is part of the Oracle Cluster Domain architecture, are deprecated.

Date: May 2021

Deprecating certain clustering features with limited adoption enables 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. Member Clusters were deprecated in Oracle Grid Infrastructure 19c. The DSC continues to be available to provide services to production clusters. However, with most of those services no longer requiring the DSC for hosting, installation of DSCs are deprecated, and can be desupported in a future release. Oracle recommends that you use any cluster or system of your choice for services previously hosted on the DSC, if applicable. Oracle will continue to support the DSC for hosting shared services, until each service can be used on alternative systems.

Deprecation of Enterprise User Security (EUS) User Migration Utility

Enterprise User Security (EUS) User Migration Utility (UMU) is deprecated in Oracle Database 21c. Use EUS Manager (EUSM) features instead.

Date: May 2021

Because organizational directory services already have records of all employees, there is no need to bulk-migrate database users to directory services. EUS Manager (EUSM) has many of the same functions as the EUS UMU. Oracle recommends that you use it place of EUS UMU.

Logical Standby and New Data Types

If you use logical standby (when not used as part of `DBMS_ROLLING`), then you can use only data types added before Oracle Database 12c Release 2 (12.2)

Date: May 2021

New data types added after Oracle Database 12c Release 1 (12.1) are not supported with Oracle Data Guard logical standby. For example, Oracle Data Guard logical standby does not support long identifiers, complex Abstract Data Types (ADTs), and spatial data types. Note that this limitation does not exist with an Oracle Data Guard physical standby database, `DBMS_ROLLING`, or Oracle GoldenGate. To obtain the benefits of a standby database with more recent data types, Oracle recommends that you consider using either a physical standby database, a snapshot standby database, or that you use the logical replication features of Oracle GoldenGate.

Deprecation of Sharded Queues

AQ sharded queues are deprecated in Oracle Database 21c. Use Transactional Event Queues (TEQ) instead.

Date: May 2021

Starting with Oracle Database 21c, AQ sharded queues are being repackaged as Transactional Event Queues (TEQ). In the Oracle Database 21c release, TEQs coexist with AQ sharded queues. However, AQ sharded queues will be desupported in a future release and will be replaced by TEQ. Oracle recommends that you move to TEQs for higher throughput and better performance on Oracle Real Application Clusters (Oracle RAC).

Deprecation of MySQL Client Library Driver for Oracle

The MySQL Client Library Driver for Oracle is deprecated in Oracle Database 21c.

Date: May 2021

The MySQL Client library driver, `liboramysql`, is deprecated. Oracle may desupport `liboramysql` in a future release. There is no replacement. This deprecation does not affect the

ability of older Oracle Database Client releases that use `liboramysql` to connect to the database. However, it is possible that the features available to use through these clients eventually can be limited.

Deprecation of TLS 1.0 and 1.1 Transport Layer Security

Starting with Oracle Database 21c, Transport Layer Security protocol versions 1.0 and 1.1 (TLS 1.0 and TLS 1.1) are deprecated.

Date: May 2021

In accordance with security best practices, Oracle has deprecated the use of TLS 1.0 and TLS 1.1. To meet your security requirements, Oracle strongly recommends that you use TLS 1.2 instead.

Deprecation of Unix Crypt (or MD5crypt) Password Verifier

The Unix Crypt (`MD5crypt`) password verifier algorithm is deprecated in Oracle Database 21c server and clients, and passwords using this algorithm will stop working in a future release.

Date: May 2021

Enterprise User Security (EUS) customers with enterprise users in Oracle Internet Directory (OID) potentially can be using older, less secure password verifiers generated by Unix Crypt, either by OID, or by the operating system, before they were migrated to OID. Unix Crypt is a less secure algorithm to hash passwords. When `MD5crypt` is removed, Oracle Database can no longer authenticate EUS or OID users with the Unix Crypt passwrod verifier type. Oracle recommends that you reset passwords in OID now, using newer, more secure hashing algorithms.

Deprecation of ODP.NET OracleConfiguration.DirectoryType Property

The Oracle Data Provider for .NET `OracleConfiguration DirectoryType` property and .NET configuration file `DIRECTORY_TYPE` setting are deprecated in Oracle Database 21c, and can be desupported in a future release.

Date: May 2021

The `OracleConfiguration DirectoryServerType` property replaces the `DirectoryType` property. The .NET configuration file `DIRECTORY_SERVER_TYPE` setting replaces the `DIRECTORY_TYPE` setting. All these properties have identical functionality. Oracle recommends to developers that you use and migrate to the new properties. The `DirectoryServerType` and `DIRECTORY_SERVER_TYPE` names better align with the `ldap.ora` parameter, `DIRECTORY_SERVER_TYPE`, which provides equivalent functionality.

Deprecation of Weaker Encryption Key Strengths

The use of weaker encryption keys is deprecated in Oracle Database 21c.

Date: May 2021

The security strength of the cipher algorithms have been changed in Oracle Database 21c with the introduction of the newest RSA BSAFE Micro Edition Suite (MES) v 4.5. The following cipher algorithms are deprecated:

- For FIPS mode

  - The FIPS default protect strength of 80 has been deprecated. This strength is still available, but will not be the default protect strength in the future. The new default protect strength for FIPS mode will be 112.
    * When the default FIPS protect strength changes from 80 to 112 with a later release, you can still revert to using the older, less secure FIPS protect strength 80 by setting a parameter.
  - Diffie Hellman and Digital Signature Algorithm (DH/DSA) with 1024 key size is deprecated. The new minimum supported key size will be 2048. The 1024 key size support will remain available when the default protect strength will be changed to 112 bits of security strength (equivalent to 2048 key size), the process strength remains at 80 bits of security strength (equivalent to 1024 key size).
- For non-FIPS mode
  - Both protect and process strength 0 (RSA key length 512) are deprecated. By default, both protect and process strength are now 80. Protect and process strength 0 (RSA key 512 and equivalent) is still available, but not recommended for use.

Related Topics

- Managing Deprecated Weaker Algorithm Keys

Deprecation of DBSNMP Packages for Adaptive Thresholds Feature

DBSNMP PL/SQL packages associated with the Adaptive Thresholds feature are deprecated in Oracle Database 21c.

Date: May 2021

Beginning with Oracle Enterprise Manager Cloud Control 13.5, all features of Database Server Adaptive Thresholds and Baseline Metric Thresholds will be removed for all Oracle Database targets. The following database server side packages that support this feature will no longer be available: `DBSNMP.BSLN`, `DBSNMP.BSLN_INTERNAL` and `DBSNMP.MGMT_RESPONSE`.

For more information about this deprecation, refer to My Oracle Support 2697846.1

Related Topics

- My Oracle Support Note 2697846.1

Deprecation of Oracle GoldenGate Replication for Oracle Sharding High Availability

Oracle GoldenGate replication support for Oracle Sharding High Availability is deprecated in Oracle Database 21c.

Date: May 2021

Deprecation of Anonymous Cipher Suites with Outbound TLS Connections

Anonymous cipher suites for outbound TLS connections (Database to other services) are deprecated with Oracle Database 21c.

Date: May 2021

Anonymous cipher suites only encrypt the connection between client and server without authenticating either party leaving it vulnerable to a person-in-the-middle attack. The following three anonymous cipher suites are deprecated with this release:

- `NZTLS_DH_ANON_WITH_AES_256_GCM_SHA384`

- NZTLS_DH_ANON_WITH_AES_128_GCM_SHA256
- NZSSL_DH_ANON_WITH_3DES_EDE_CBC_SHA

To provide better protection for your connections, Oracle recommends using the strongest possible non-anonymous TLS cipher suites.

Deprecation of the KERBEROS5PRE Adapter

The use of the KERBEROS5PRE adapter is deprecated with Oracle Database 21c. Oracle recommends that you use the KERBEROS5 adapter instead.

Date: May 2021

The KERBEROS5 adapter is the primary adapter supported by Oracle for Kerberos authentication. KERBEROS5PRE is not needed anymore, and will be desupported in a future release. This change simplifies Kerberos configuration.

Deprecation of Oracle Wallet Manager

Oracle Wallet Manager (OWM) is deprecated with Oracle Database 21c.

Date: May 2021

Instead of using Oracle Wallet Manager, Oracle recommends that you use the command line tool orapki .

Deprecation of Oracle Enterprise Manager Database Express

Oracle Enterprise Manager Database Express (EM Express) is deprecated, and will be removed in a future Oracle Database release.

Date: May 2021

EM Express is a web-based database management tool that is built inside the Oracle Database. It supports key performance management and basic database administration functions. Many of EM Express's capabilities are also available in Oracle SQL Developer, which is included in all Oracle Database editions. Oracle recommends that you replace your use of EM Express with Oracle SQL Developer.

Deprecation of Service Attribute Value SESSION_STATE_CONSISTENCY = STATIC

The combination of session attribute values FAILOVER_TYPE = TRANSACTION with SESSION_STATE_CONSISTENCY = STATIC is deprecated in Oracle Database 21c.

Date: May 2023

The use of session attribute values FAILOVER_TYPE = TRANSACTION with SESSION_STATE_CONSISTENCY = STATIC is no longer a supported service attribute combination.

Instead use one of the following combinations in the service configuration:

- FAILOVER_TYPE = AUTO with SESSION_STATE_CONSISTENCY = AUTO
- FAILOVER_TYPE = TRANSACTION with SESSION_STATE_CONSISTENCY = DYNAMIC

These configurations enforce session state tracking in Oracle Database, ensuring that session state is preserved at session migration and session failover. Note that starting with Oracle Database 21c, you may also wish to set the RESET_STATE attribute to clear your session state

set by applications in request at the end of the request. For more information, see `RESET_STATE`.

Deprecation of SHA-1 use for SQLNET and DBMS_CRYPTO

The use of SHA-1 with `DBMS_CRYPTO`, `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` and `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` is deprecated.

May 2021

Using SHA-1 (Secure Hash Algorithm 1) with the parameters `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` and `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` is deprecated in this release, and can be desupported in a future release. Using SHA-1 ciphers with `DBMS_CRYPTO` is also deprecated (`HASH_SH1`, `HMAC_SH1`). Instead of using SHA1, Oracle recommends that you start using a stronger SHA-2 cipher in place of the SHA-1 cipher.

Deprecation of ACFSUTIL REPL REVERSE

The `acfsutil repl reverse` command is deprecated in Oracle Database 21c. Use `repl failover` or `repl switchover` instead.

Date: August 2021

The Oracle Advanced Cluster File System (ACFS) command utility `acfsutil` includes the commands `repl failover` and `repl switchover`. These commands provide more functionality, including all the functions of `acfsutil repl reverse`. For this reason, Oracle is deprecating the `acfsutil repl reverse` command.

Deprecation of Oracle OLAP

Analytic workspaces, the OLAP DML programming language, and the OLAP Java API are deprecated in Oracle Database 21c.

Date: August 2021

For new applications requiring advanced analytic capabilities, Oracle recommends that you consider analytic views (a feature of Oracle Database), or Oracle Essbase. Oracle analytic views are a feature of every Oracle Database edition. If your application uses OLAP for dimensional query and reporting applications, then Oracle recommends that you consider Oracle analytic views as a replacement for OLAP. Analytic views provide a fast and efficient way to create analytic queries of data stored in existing database tables and views. With Oracle analytic views, you obtain a dimensional query model and supporting metadata without requiring a "cube build/update" process. The elimination of the cube build/update process relieves scalability constraints (model complexity and data volume), simplifies the data preparation pipeline, and reduces or eliminates data latency. In addition, you can use analytic views with OLTP applications, external tables, and non-relational data types (for example, JSON) when these non-relational data types are wrapped by relational views.

Deprecation of Oracle Database Extensions for .NET

Oracle Database Extensions for .NET is deprecated in Oracle Database 21c. Oracle recommends that you either place .NET code in the middle tier, or use the External Procedures feature, or rewrite the code using PL/SQL or Java.

Date: September 2021

Oracle Database Extensions for .NET is a feature of Oracle Database on Microsoft Windows that enables you to use stored procedures and functions written in a language managed by .NET, such as C#.

Oracle Database hosts the Microsoft Common Language Runtime (CLR) in an external process, outside of the Oracle Database process. Application developers can write stored procedures and functions using any .NET compliant language, such as C# and VB.NET, and use these .NET stored procedures in the database, in the same manner as other PL/SQL or Java stored procedures. .NET stored procedures can be called from PL/SQL packages, procedures, functions, and triggers; from SQL statements; or from anywhere a PL/SQL procedure or function can be called.

Migration options include:

- Moving the .NET code (assemblies) into a middle tier
- Using the External Procedures feature to have the external process load and execute the .NET assembly
- Rewriting the stored procedures using PL/SQL or Java

Deprecation of Repository Events

The use of repository events to trigger application actions is deprecated in Oracle Database 21c (21.3). There is no replacement.

October 2021

Repository events are events that can be used to trigger application actions. Repository events include repository changes, such as creating, deleting, locking, unlocking, rendering, linking, unlinking, placing under version control, checking in, checking out, unchecking out (reverting a checked out version), opening, and updating a resource. The deprecation of the use of repository events includes deprecation of the `DBMS_XEVENT` package, and the following subprogram groups:

- `XDBevent`
- `XDBRepositoryEvent`
- `XDBHandler`
- `XDBHandlerList`
- `XDBPath`
- `XDBLink`

Deprecation of Quality of Service Management

Starting with Oracle Database Release 21c, Oracle Quality of Service Management (QoSM, or QOS Management) is deprecated.

Date: November 2022

Oracle QoSM automates the workload management for an entire system by adjusting the system configuration based on predefined policies to keep applications running at the performance levels needed. Applications and databases are increasingly deployed in systems that provide some of the resource management capabilities of QoSM. At the same time, Oracle's Autonomous Health Framework has been enhanced to adjust and provide recommendations to mitigate events and conditions that impact the health and operational

capability of a system and its associated components. For those reasons, QoSM is deprecated with Oracle Database 21c.

Grid Infrastructure Management Repository Deprecation

The Grid Infrastructure Management Repository (GIMR), or Management Database (`MGMTDB`), is deprecated in Oracle Database 21c.

Date: May 2023

Solutions using the GIMR (GIMR clients) such as Oracle Autonomous Health Framework (AHF), Cluster Health Monitor (CHM), Oracle Cluster Health Advisor (CHA), or Oracle Fleet Patching and Provisioning (FPP), will automatically transition to alternative repositories as needed. As a purpose-built database used exclusively by GIMR clients, the deprecation and potential removal of the GIMR in future releases will be seamless for Oracle Grid Infrastructure and related services.

Deprecated Views in Oracle Database 21c

As part of your upgrade plan, review the views that are deprecated starting with this Oracle Database release.

- Deprecation of Traditional Auditing Views
  As a result of deprecating traditional auditing, the views associated with traditional auditing are also deprecated.

Deprecation of Traditional Auditing Views

As a result of deprecating traditional auditing, the views associated with traditional auditing are also deprecated.

Date: May 2021

- Static data dictionary views:
  - `ALL_DEF_AUDIT_OPTS`
  - `AUDIT_ACTIONS`
  - `DBA_AUDIT_EXISTS`
  - `DBA_AUDIT_OBJECT`
  - `DBA_AUDIT_SESSION`
  - `DBA_AUDIT_STATEMENT`
  - `DBA_AUDIT_TRAIL`
  - `DBA_COMMON_AUDIT_TRAIL`
  - `DBA_FGA_AUDIT_TRAIL`

  - DBA_OBJ_AUDIT_OPTS
  - DBA_PRIV_AUDIT_OPTS
  - DBA_STMT_AUDIT_OPTS
  - USER_AUDIT_OBJECT
  - USER_AUDIT_SESSION
  - USER_AUDIT_STATEMENT
  - USER_AUDIT_TRAIL
  - USER_OBJ_AUDIT_OPTS
- Dynamic performance view:
  - V$XML_AUDIT_TRAIL

Deprecated Parameters in Oracle Database 21c

As part of your upgrade plan, review the initialization parameters that are deprecated starting with this Oracle Database release.

- Deprecation of Traditional Auditing Initialization Parameters
  As a result of deprecating traditional auditing, the initialization parameters associated with traditional auditing are also deprecated.

Deprecation of Traditional Auditing Initialization Parameters

As a result of deprecating traditional auditing, the initialization parameters associated with traditional auditing are also deprecated.

Date: May 2021

- Initialization parameters:
  - AUDIT_FILE_DEST
  - AUDIT_SYS_OPERATIONS
  - AUDIT_SYSLOG_LEVEL
  - AUDIT_TRAIL

Oracle Database 19c Behavior Changes, Desupports, and Deprecations

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.
- Desupported Features in Oracle Database 19c
  As part of your upgrade plan, review the features that are desupported in Oracle Database 19c.
- Desupported Parameters in Oracle Database 19c
  As part of your upgrade plan, review the initialization parameters that are not supported starting with Oracle Database 19c.
- Deprecated Features in Oracle Database 19c
  As part of your upgrade plan, review the features that are deprecated in Oracle Database 19, 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 Oracle Database 19c.

Behavior Changes for Oracle Database 19c Upgrade Planning

Review these behavior changes to help plan for upgrades to Oracle Database 19c.

Behavior changes can include default changes for parameters or features, product name changes, and other changes to the database configuration that may require your attention.

- All Time Zone Files (DST) Included in Release Updates (RUs)
  Starting with Oracle Database 19c RU 19.18.0, all available DST patches are installed with the RU, and deployed into the *Oracle_home*/oracore/zoneinfo directory.
- 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
  In this release, you can install Oracle Database Standard Edition 2 in high availability mode.

- Manage "Installed but Disabled" Module Bug Fixes with DBMS_OPTIM_BUNDLE
  To manage the implementation of Oracle Database bug fixes that cause a SQL execution plan change, use `DBMS_OPTIM_BUNDLE`.
- Windows Authentication No Longer Uses NTLM by Default
  For Microsoft Windows installations with `AUTHENTICATION_SERVICES=NTS`, in this Oracle Database release, the `SQLNET.NO_NTLM` parameter setting in the `sqlnet.ora` file defaults to `TRUE`, which can cause ORA-12638 errors.

All Time Zone Files (DST) Included in Release Updates (RUs)

Starting with Oracle Database 19c RU 19.18.0, all available DST patches are installed with the RU, and deployed into the *Oracle_home*`/oracore/zoneinfo` directory.

Installing DST patches does not affect current database operation. However, installing the patches with the RU makes it easier for you to adjust the timezone version of your database, if you have a requirement to do so. For example, if you are using Transportable Tablespaces, or Full Transportable Export/Import, then you must ensure that your source and target databases are using identical character sets and identical time zone settings. With this change, you can more easily choose to change your destination database to use a different time zone file version than the default.

> **Note:**
>
> By default, AutoUpgrade changes the database time zone to the latest available level. If you don't want the time zone to be upgraded, then you must explicitly set the local parameter `timezone_upg` in your AutoUpgrade configuration file to `no`. For example:
>
> ```
> upg1.timezone_upg=no
> ```
>
> When you patch a database with RU 19.18 or later, a new database created with Database Configuration Assistant (DBCA) in that patched Oracle home will be created with the latest time zone files.

Related Topics

- Choosing a Time Zone File
- RUs contain now all available DST patches
- Primary Note DST FAQ : Updated DST Transitions and New Time Zones in Oracle RDBMS and OJVM Time Zone File Patches (Doc ID 412160.1)

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

In this release, you can install Oracle Database Standard Edition 2 in high availability mode.

Standard Edition High Availability provides cluster-based failover for single-instance 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 Advanced 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.

Standard Edition High Availability is supported on Linux x86-64, Microsoft Windows, and HP-UX Itanium.

> **Note:**
>
> This section is specific to Standard Edition High Availability, which provides cluster-based database failover for Standard Edition Oracle Databases 21c and later. For more information about high availability options for Oracle Database, see *Oracle Clusterware Administration and Deployment Guide*.

Manage "Installed but Disabled" Module Bug Fixes with DBMS_OPTIM_BUNDLE

To manage the implementation of Oracle Database bug fixes that cause a SQL execution plan change, use `DBMS_OPTIM_BUNDLE`.

After you upgrade your database, the bug fix patches that can cause execution plan changes included in the Release Updates are installed disabled by default. These bug fixes will not be activated until you enable the fixes. You can either enable these fixes manually, or use the `DBMS_OPTIM_BUNDLE` package.

Oracle strongly recommends that you enable these disabled patches that you want to use in your production system, and run complete workload performance tests using these patches as part of your upgrade test plan.

For more information about using `DBMS_OPTIM_BUNDLE` to enable patches that were disabled because they can change execution plans, see *Oracle Database PL/SQL Packages and Types Reference*, and My Oracle Support note 2147007.1.

Related Topics

- DBMS_OPTIM_BUNDLE
- My Oracle Support Doc ID 2147007.1 Managing "installed but disabled" bug fixes in Database Release Updates using DBMS_OPTIM_BUNDLE

Windows Authentication No Longer Uses NTLM by Default

For Microsoft Windows installations with `AUTHENTICATION_SERVICES=NTS`, in this Oracle Database release, the `SQLNET.NO_NTLM` parameter setting in the `sqlnet.ora` file defaults to `TRUE`, which can cause ORA-12638 errors.

Date: August 2023

In previous releases, the default for `SQLNET.NO_NTLM` was `FALSE`. `SQLNET.NO_NTLM` controls whether `NTLM` can be used with NTS authentication (`AUTHENTICATION_SERVICES=NTS`). A `TRUE` setting means that NTLM cannot be used in NTS authentication. Because NTLM does not

normally provide mutual authentication and is hence less secure, a `TRUE` setting for `SQLNET.NO_NTLM` makes the database and client more secure.

The `SQLNET.NO_NTLM` parameter is used on both the server and the client. If you have upgraded a Microsoft Windows installation of Oracle Database, or upgraded a client in which `SQLNET.NO_NTLM` had not been set, then its default will be `TRUE`. In that case, when you have `SQLNET.AUTHENTICATION_SERVICES=NTS` in your `sqlnet.ora`, clients can encounter the error `ORA-12638: Credential retrieval failed`.

If you prefer to use NTLM authentication for certain clients, then set this parameter as required in client-side `sqlnet.ora` files:

```
SQLNET.NO_NTLM=FALSE
```

You must include this setting on both the server and client, and this setting should be the same on both. Ideally, you should ensure that `SQLNET.NO_NTLM` is set to `TRUE`. However, if there is an authentication failure in `extproc`, a virtual account, or a local account on Windows, set the client `SQLNET.NO_NTLM` to `FALSE`, and then retry the login. If you change `SQLNET.NO_NTLM` on the server, then you must restart the database.

Related Topics

- Upgrade Your Database Now - ORA-12638 on Windows only from Oracle 19.10.0 onwards
- Windows : While Connect to Database Getting ORA-12638 After Applying Jan 2021 WINDBBP 19.10.0.0.210119 On Windows 64/32 Bit Database Client (Doc ID 2757734.1)

Desupported Features in Oracle Database 19c

As part of your upgrade plan, review the features that are desupported in Oracle Database 19c.

- Desupport of Oracle Data Provider for .NET Promotable Transaction Setting
- Desupport of Oracle Multimedia
- 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 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.
Date: April 2019

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.
Date: April 2019

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.

Date: April 2019

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.

Date: April 2019

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.

Date: April 2019

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.

Date: April 2019

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.

Date: April 2019

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.

Date: April 2019

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).

Date: April 2019

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

Desupported Parameters in Oracle Database 19c

As part of your upgrade plan, review the initialization parameters that are not supported starting with Oracle Database 19c.

- 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.

Date: April 2019

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.

Date: April 2019

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.

Date: November 2019

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.

Date: September 2020

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.

Deprecated Features in Oracle Database 19c

As part of your upgrade plan, review the features that are deprecated in Oracle Database 19, and review alternatives for your application strategies.

- Deprecation of URL_DATASTORE Text Type
  Starting with Oracle Database 19c, the Oracle Text type URL_DATASTORE is deprecated. Use NETWORK_DATASTORE instead.
- Deprecation of FILE_DATASTORE Type
  Starting with Oracle Database 19c, the Oracle Text type FILE_DATASTORE is deprecated. Use DIRECTORY_DATASTORE instead.
- 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 Database Vault role `DV_REALM_OWNER` role is deprecated with no replacement.
- Deprecation of DV_REALM_RESOURCE Role
  The Oracle Database Vault role `DV_REALM_RESOURCE` is deprecated with no replacement.
- Deprecation of DV_PUBLIC Role
  The Oracle Database 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 OLAP
  Analytic workspaces, the OLAP DML programming language, and the OLAP Java API are deprecated in Oracle Database 19c.
- 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.
- Deprecation of OPatch and OPatchAuto for Out-of-Place Patching
  Oracle recommended out-of-place patching using Opatch and OPatchAuto, but Oracle is transitioning to a simpler and easier Gold image process for out-of-place patching starting with Oracle Database 23ai On Premises.
- Data Recovery Advisor (DRA) Deprecation
  Starting in Oracle Database 19c, the Data Recovery Advisor (DRA) feature is deprecated.
- DBUA and Manual Upgrade Deprecated
  Oracle recommends using AutoUpgrade for database upgrades.
- Cluster Time Synchronization Service Deprecation
  Cluster Time Synchronization Service (CTSS) is deprecated in Oracle Database 19c.
- ACFS Deprecation on IBM AIX
  The ACFS storage option on IBM AIX is deprecated in Oracle Database 19c.
- RECOVER…SNAPSHOT TIME Deprecated
  The `RECOVER…SNAPSHOT TIME` method of recovering a database to a point in time using a particular snapshot is deprecated in Oracle Database 19c.
- Oracle Data Masking and Subsetting with Oracle Real Application Testing Deprecated
  The integration of data masking with Oracle Real Application Testing is deprecated in Oracle Database 19c.

Deprecation of URL_DATASTORE Text Type

Starting with Oracle Database 19c, the Oracle Text type `URL_DATASTORE` is deprecated. Use `NETWORK_DATASTORE` instead.

Date: April 2019

The `URL_DATASTORE` type is used for text stored in files on the internet (accessed through HTTP or FTP), and for text stored in local file system files (accessed through the file protocol). It is replaced with `NETWORK_DATASTORE`, which uses ACLs to allow access to specific servers. This change aligns Oracle Text more closely with the standard operating and security model for accessing URLs from the database.

Deprecation of FILE_DATASTORE Type

Starting with Oracle Database 19c, the Oracle Text type `FILE_DATASTORE` is deprecated. Use `DIRECTORY_DATASTORE` instead.

Date: April 2019

Oracle recommends that you replace `FILE_DATASTORE` text indexes with the `DIRECTORY_DATASTORE` index type, which is available starting with Oracle Database 19c. `DIRECTORY_DATASTORE` provides greater security because it enables file access to be based on directory objects.

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`.

Date: April 2019

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.

Date: April 2019

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).

Date: April 2019

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.

Date: April 2019

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.

Date: April 2019

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.

Date: April 2019

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.

Date: April 2019

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.

Date: April 2019

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 Database Vault role `DV_REALM_OWNER` role is deprecated with no replacement.

Date: April 2019

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 Database Vault role `DV_REALM_RESOURCE` is deprecated with no replacement.

Date: April 2019

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 Database Vault role `DV_PUBLIC` role is deprecated with no replacement.

Date: April 2019

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.

Date: April 2019

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 OLAP

Analytic workspaces, the OLAP DML programming language, and the OLAP Java API are deprecated in Oracle Database 19c.

Date: April 2019

This deprecation is announced in Oracle Database 19c and Oracle Database 21c. For new applications requiring advanced analytic capabilities, Oracle recommends that you consider analytic views (a feature of Oracle Database), or Oracle Essbase. Oracle analytic views are a feature of every Oracle Database edition. If your application uses OLAP for dimensional query and reporting applications, then Oracle recommends that you consider Oracle analytic views as a replacement for OLAP. Analytic views provide a fast and efficient way to create analytic queries of data stored in existing database tables and views. With Oracle analytic views, you obtain a dimensional query model and supporting metadata without requiring a "cube build/update" process. The elimination of the cube build/update process relieves scalability

constraints (model complexity and data volume), simplifies the data preparation pipeline, and reduces or eliminates data latency. In addition, you can use analytic views with OLTP applications, external tables, and non-relational data types (for example, JSON) when these non-relational data types are wrapped by relational views.

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.

Date: April 2019

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.

Date: October 2019

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

Date: October 2019

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.

Date: October 2019

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.

Date: October 2019

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.

Date: October 2019

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. This deprecation applies to clustered deployments, including Oracle Real Application Clusters and Oracle RAC One Node.

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.

Date: May 2020

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.

Deprecation of OPatch and OPatchAuto for Out-of-Place Patching

Oracle recommended out-of-place patching using Opatch and OPatchAuto, but Oracle is transitioning to a simpler and easier Gold image process for out-of-place patching starting with Oracle Database 23ai On Premises.

Date: May 2023

Oracle continues to recommend out-of-place patching for the Oracle Database and Oracle Grid Infrastructure homes for Oracle Database 19c and 21c. However, Oracle is transitioning to a streamlined process that uses Gold Images for out-of-place patching of Oracle Database. In addition, Oracle also provides the fully automated Fleet Patching and Provisioning solution for patching your software. Accordingly, Oracle is announcing the deprecation of using Opatch and OpatchAuto for out-of-place patching. This deprecation does not affect in-place patching.

For Oracle Grid Infrastructure, Oracle recommends that you use `gridSetup.sh` with the `switchGridHomes` option. For standalone database homes, Oracle recommends that you use a fresh installation, and perform an out-of-place patch using Oracle Universal Installer with the `applyRU` and `applyOneOffs` options. You can then patch your databases either with AutoUpgrade, or with Fleet Patching and Provisioning, which automates the entire patch process for you.

Data Recovery Advisor (DRA) Deprecation

Starting in Oracle Database 19c, the Data Recovery Advisor (DRA) feature is deprecated.

Date: July 2023

The deprecation of DRA includes deprecation of the following Oracle Recovery Manager (RMAN) commands: `LIST FAILURE`, `ADVISE FAILURE`, `REPAIR FAILURE`, and `CHANGE FAILURE`. Database administrators will no longer have access to these commands. There is no replacement feature for DRA.

DBUA and Manual Upgrade Deprecated

Oracle recommends using AutoUpgrade for database upgrades.

Date: July 2023

To improve the reliability and support for Oracle Database upgrades, Oracle is deprecating Database Upgrade Assistant (DBUA), and manual upgrades using the Parallel Upgrade Utility (`catctl.pl`) , and the database upgrade scripts `dbupgrade` and `dbupgrade.cmd` in Oracle Database 19c. Use AutoUpgrade for Oracle Database upgrades. These obsolete upgrade methods can be removed in a future release update. Deprecating obsolete upgrade methods enables Oracle to focus on improving and extending the features and manageability of the AutoUpgrade utility.

Cluster Time Synchronization Service Deprecation

Cluster Time Synchronization Service (CTSS) is deprecated in Oracle Database 19c.

Date: January 2024

To synchronize time between cluster member nodes, use either an operating system configured network time protocol such as `ntp` or `chrony`, or Microsoft Windows Time service. To verify that you have network time synchronization configured, you can use the `cluvfy comp clocksync -n all` command.

ACFS Deprecation on IBM AIX

The ACFS storage option on IBM AIX is deprecated in Oracle Database 19c.

Date: December 2024

To align the support for the IBM AIX platform, Oracle Automatic Storage Management Cluster File System (ACFS) on IBM AIX (called Oracle Advanced Cluster File System starting with Oracle Database 21c) is deprecated in Oracle Database 19c. For data or files currently stored on an ACFS file system that cannot be stored in the Oracle database, the IBM General Parallel File System (GPFS) can be used.

RECOVER…SNAPSHOT TIME Deprecated

The `RECOVER…SNAPSHOT TIME` method of recovering a database to a point in time using a particular snapshot is deprecated in Oracle Database 19c.

Date: February 2024

Instead of `RECOVER…SNAPSHOT TIME`, Oracle recommends that you use `ALTER DATABASE BEGIN/END BACKUP` before and after creating the storage snapshot of the data files and then use `RECOVER .. UNTIL TIME` to a specific timestamp or system change number (SCN) after the `END BACKUP` completion time. Oracle recommends that `ALTER DATABASE BEGIN/END BACKUP` always be used when performing snapshots on a running database to ensure data recovery integrity. Archived log redo logs must be separately backed up and restored for recovery operations.

Note that Oracle's best practice for database backup and recovery is to use Recovery Manager (RMAN) only, not storage snapshots, and is integrated with Zero Data Loss Recovery Appliance (ZDLRA) to offer the highest levels of database protection.

Related Topics

- IT Infrastructure Engineered Systems Zero Data Loss Recovery Appliance

Oracle Data Masking and Subsetting with Oracle Real Application Testing Deprecated

The integration of data masking with Oracle Real Application Testing is deprecated in Oracle Database 19c.

Date: February 2024

For testing, Oracle recommends that you keep the test database in the production realm for security, but create user accounts for testing that cannot access your data to run both SQL Performance Analyzer (SPA) trials and Database Replay.

Deprecated Initialization Parameters in Oracle Database 19c

As part of your upgrade plan, review the initialization parameters that are deprecated in Oracle Database 19c.

- 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)

Date: April 2019

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.

Date: April 2019

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.

Date: November 2019

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.

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

- 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.
- 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.
- Deprecated Features in Oracle Database 18c
  Review the deprecated features listed in this section to prepare to use alternatives after you upgrade.

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-21c` 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, run AutoUpgrade in analyze mode. 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*

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, `BFILE`s, 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`.

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.

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.
- Deprecation of Oracle Multimedia
  Starting in Oracle Database 18c, Oracle Multimedia is deprecated. Oracle Multimedia will be desupported in 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.

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.

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.

Oracle Database 12c Release 2 (12.2) Behavior Changes, Desupports, and Deprecations

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)
- Desupported Features in Oracle Database 12c Release 2 (12.2)
  Review this list of desupported features as part of your upgrade planning.
- 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.
- 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.
- 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.

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
  Learn how you can use Oracle Fleet Patching and Provisioning (Oracle FPP) to provision your Oracle homes, and to manage your software lifecycle.
- 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 12c, if you use Unified Auditing, then you can have two AUDSYS roles in your Oracle Database 11g 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 12c release 1 (12.1) or later.

If you have created `AUDIT_ADMIN` and `AUDIT_VIEWER` users or roles with Oracle Database 12c 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 12c 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.

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.

> **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 and Offline 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

Learn how you can use Oracle Fleet Patching and Provisioning (Oracle FPP) to provision your Oracle homes, and to manage your software lifecycle.

> **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 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.

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 releases are 12.2 and later releases. 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) and later releases.

  – Installations running without Oracle Grid Infrastructure.

  The Oracle Fleet Patching and Provisioning Server can provision new installations, and 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, processed 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*

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.

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 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).

> **Note:**
>
> The Pre-Upgrade Information Tool is now replaced with the AutoUpgrade Utility, which can assist you with upgrades from Oracle Database 11.2.0.4 and later releases. Download the latest available version.

Related Topics

- My Oracle Support note 2485457.1

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)

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_DATABASE`initialization 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*

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.

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*

Oracle Database 12c Release 1 (12.1) Behavior Changes, Desupports, and Deprecations

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)
- Desupported Features in Oracle Database 12c Release 1 (12.1)
  Review this list of desupported features as part of your upgrade planning.
- 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.
- Deprecated Initialization Parameters in Oracle Database 12c Release 1 (12.1)
  As part of your upgrade plan, review the initialization parameters that are deprecated in Oracle Database 12c Release 1 (12.1).
- 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.

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.
- 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).
- 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
```

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

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

Desupported Features in Oracle Database 12c Release 1 (12.1)

Review this list of desupported features as part of your upgrade planning.

- 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 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 CLEANUP_ORACLE_BASE Property
  In Oracle Database 12*c*, the `CLEANUP_ORACLE_BASE` property is removed for response file (silent) deinstalls.
- 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 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 Net Listener Password
  In Oracle Database 12c, the Oracle Net Listener password feature 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 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 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 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
- Desupported Features on Microsoft Windows Platforms
  Review these topics to learn which features are deprecated or desupported with Oracle Database 12*c* for Microsoft Windows.

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 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 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.

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 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 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 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 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 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 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*

Desupported Features on Microsoft Windows Platforms

Review these topics to learn which features are deprecated or desupported with Oracle Database 12*c* for Microsoft Windows.

- 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.
- 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 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

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

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 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.pl`instead 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.

- 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.
- Deprecated Default Realms for Oracle Database Vault
  The Oracle Data Dictionary realm and Oracle Enterprise Manager realm are deprecated in this release.
- Deprecated Default Rule Sets for Oracle Database Vault
  Rule sets listed here are deprecated with Oracle Database 12c Release 1.
- 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.
- Deprecated EndToEndMetrics-related APIs
  `EndToEndMetrics` -related APIs are deprecated in Oracle Database 12*c*. Use Universal Connection Pool instead.
- Deprecation of Oracle Restart
  Oracle Restart, also known as Oracle Grid Infrastructure for a Standalone Server, is deprecated in Oracle Database 12*c*.
- Deprecation of -checkpasswd for QOSCTL Quality of Service (QoS) Command
  The syntax `qosctl -checkpasswd username password` 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.
- 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
- 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.
- Oracle Data Provider for .NET Deprecated Programming Interfaces
  Oracle Data Provider for .NET application programming interfaces for Transaction Guard are deprecated in Oracle Database 12c (12.1.0.2), and can be desupported in a future release.
- 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 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 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.
- Oracle Data Pump Export Utility Features
  The `XML_CLOBS` option of the Oracle Data Pump Export `DATA_OPTIONS` parameter is deprecated.
- 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).
- Deprecated defineColumnType Method
  The JDBC method `defineColumnType` is deprecated in Oracle Database 12*c* Release 1.
- 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.
- 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
- Deprecation of Stored List of Administrative Users for Cluster Administration
  Cluster administration is managed differently starting with Oracle Database 12*c*.
- Deprecation of -cleanupOBase
  The `-cleanupOBase` flag of the deinstallation tool is deprecated in Oracle Database 12c Release 1 (12.1).
- 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 Advanced Replication
  Oracle Database Advanced Replication is deprecated in Oracle Database 12c.
- 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.
- Deprecation of JPublisher

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

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

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 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*

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.

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

Deprecation of Oracle Restart

Oracle Restart, also known as Oracle Grid Infrastructure for a Standalone Server, is deprecated in Oracle Database 12*c*.

Oracle Restart is restricted to manage single-instance Oracle databases and Oracle ASM instances only, and is subject to desupport in future releases. Oracle continues to provide Oracle ASM as part of the Oracle Grid Infrastructure installation for Standalone and Cluster deployments, and Oracle continues to provide Oracle RAC One Node.

See Also:

- My Oracle Support Note 1584742.1 for more information about Oracle Restart deprecation announcement and its replacement:

  https://support.oracle.com/epmos/faces/DocumentDisplay?id=1584742.1&displayIndex=1

- *Oracle Database Administrator's Guide*

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

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 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`

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.

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

Oracle Data Provider for .NET Deprecated Programming Interfaces

Oracle Data Provider for .NET application programming interfaces for Transaction Guard are deprecated in Oracle Database 12c (12.1.0.2), and can be desupported in a future release.

The following application programming interfaces are deprecated:

- `OracleLogicalTransactionStatus` class
- `OracleConnection.LogicalTransactionId` property
- `OracleConnection.GetLogicalTransactionStatus` method

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 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 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.

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

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

Deprecated defineColumnType Method

The JDBC method `defineColumnType` is deprecated in Oracle Database 12*c* Release 1.

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 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`

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 -cleanupOBase

The `-cleanupOBase` flag of the deinstallation tool is deprecated in Oracle Database 12c Release 1 (12.1).

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.

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*

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.

Deprecation of JPublisher

Oracle JPublisher is deprecated in Oracle Database 12*c* Release 1, as of October 2014, and all JPublisher features are desupported and unavailable in Oracle Database 12*c* Release 2. Oracle recommends that you use the alternatives listed here:

- To continue to use Web service callouts, Oracle recommends that you use the OJVM Web Services Callout utility, which is a replacement for the Web Services Callout utility.
- To create Java client applications for PL/SQL programs and SQL objects, Oracle recommends that developers use other JDK development tools that assist you to create java `STRUCT` classes, and other prestructured options.

> **See Also:**
>
> My Oracle Support Note 1937939.1 for more information about JPublisher deprecation and desupport: https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1937939.1
>
> Also, see JDK Tools and Utilities on Oracle Technology Network:
>
> http://docs.oracle.com/javase/8/docs/technotes/tools/

Deprecated Initialization Parameters in Oracle Database 12c Release 1 (12.1)

As part of your upgrade plan, review the initialization parameters that are 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.
- 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*.
- LOG_ARCHIVE_LOCAL_FIRST Initialization Parameter Desupported
  The `LOG_ARCHIVE_LOCAL_FIRST` initialization parameter is removed and desupported in Oracle Database 12*c*.

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

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

LOG_ARCHIVE_LOCAL_FIRST Initialization Parameter Desupported

The `LOG_ARCHIVE_LOCAL_FIRST` initialization parameter is removed and desupported in Oracle Database 12*c*.

`LOG_ARCHIVE_LOCAL_FIRST` specified when the archiver processes (`ARCn`) transmit redo data to remote standby database destinations. It was deprecated in Oracle Database 11g.

Oracle recommends that you use `LOG_ARCHIVE_DEST_n`. Set the attributes for the `LOG_ARCHIVE_DEST_n` initialization parameter to control different aspects of how redo transport services transfer redo data from a production or primary database destination to another (standby) database destination.

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

A

AutoUpgrade REST APIs

The AutoUpgrade REST APIs for Oracle Database enable you to automate database upgrades using RESTful Services in a web environment.

- Introduction to AutoUpgrade REST APIs
  Learn about the AutoUpgrade Utility REST API method of upgrading Oracle Database.
- Get Started
  Set up your system so that you upgrade your databases using the AutoUpgrade REST APIs.
- REST APIs for AutoUpgrade
  These REST APIs are available for the AutoUpgrade REST services feature in Oracle REST Data Services (ORDS).

Introduction to AutoUpgrade REST APIs

Learn about the AutoUpgrade Utility REST API method of upgrading Oracle Database.

- About AutoUpgrade REST APIs
  The AutoUpgrade REST API implementation uses the Oracle REST Data Services plug-in framework.

About AutoUpgrade REST APIs

The AutoUpgrade REST API implementation uses the Oracle REST Data Services plug-in framework.

The AutoUpgrade REST APIs provide a uniform interface for AutoUpgrade on client-server architecture, and enable you to leverage and integrate AutoUpgrade features through HTTP and HTTPS protocols to upgrade Oracle Databases automatically. The Oracle REST Data Services implementation is based on a Java Enterprise Edition (Java EE) data service that provides enhanced security, file-caching features, and RESTful Web Services. This service provides increased flexibility for standalone deployments, as well as through servers, such as Oracle WebLogic Server and Apache Tomcat. Oracle REST Data Services is a common plugin mechanism used by most Oracle Database components and tools.

Related Topics

- REST APIs for Oracle Database

Get Started

Set up your system so that you upgrade your databases using the AutoUpgrade REST APIs.

- Install cURL
  The examples within this document use the cURL command-line tool to demonstrate how to access the Autoupgrade REST API.

- How to Set Up and Use AutoUpgrade REST APIs
  To use the AutoUpgrade REST APIs, you must install Oracle REST Data Services, and enable AutoUpgrade REST service.
- How to Use AutoUpgrade REST API Authentication Privileges
  To configure users with AutoUpgrade Administrator user privileges to access AutoUpgrade REST APIs, complete this setup procedure.

Install cURL

The examples within this document use the cURL command-line tool to demonstrate how to access the Autoupgrade REST API.

To connect securely to the server, you must install a version of cURL that supports SSL and provide an SSL certificate authority (CA) certificate file or bundle to authenticate against the Verisign CA certificate.

1. In your browser, navigate to the cURL home page, and click **Download** in the left navigation menu.
2. On the cURL Releases and Downloads page, locate the SSL-enabled version of the cURL software that corresponds to your operating system, click the link to download the tar or ZIP file, and install the software.
3. Navigate to the cURL CA certificates extracted from Mozilla page, and download the The Mozilla CA certificate store in PEM format (`cacert.pem`) in the folder where you installed cURL.
4. Open a command window, navigate to the directory where you installed cURL, and set the cURL environment variable, `CURL_CA_BUNDLE`, to the location of an SSL certificate authority (CA) certificate bundle. For example:

   ```
   C:\curl> set CURL_CA_BUNDLE=cacert.pem
   ```

You are now ready to send REST requests to the AutoUpgrade REST API instance using cURL.

Related Topics

- cURL command line tool and library
- CA certificates extracted from Mozilla

How to Set Up and Use AutoUpgrade REST APIs

To use the AutoUpgrade REST APIs, you must install Oracle REST Data Services, and enable AutoUpgrade REST service.

Before you enable or install the AutoUpgrade REST API, ensure that you have completed installation of the Oracle REST Data Services in standalone mode:

Configuring and Installing Oracle REST Data Services

1. After installing `ords.war` and before starting ORDS service, you must set the `autoupgrade.api.enabled` property to `true` (the default is `false`), and set the environment variables for the APIs.

   In the following example:

   - `autoupgrade.api.loglocation` is specified to the path `/databases/working`

- `autoupgrade.api.aulocation` specifies that the autoupgrade.jar file is set in the path `/autoupgrade/ords/ords/autoupgrade.jar`
- `autoupgrade.api.jvmlocation` specifies that the Java location is in `/usr/bin/java`.

```
java -jar ords.war set-property autoupgrade.api.enabled true
java -jar ords.war set-property autoupgrade.api.loglocation /databases/
working
java -jar ords.war set-property autoupgrade.api.aulocation
/autoupgrade/ords/ords/autoupgrade.jar
java -jar ords.war set-property autoupgrade.api.jvmlocation /usr/bin/java
```

2. Revise your AutoUpgrade configuration file to use a JSON format.

   To submit a new task through AutoUpgrade REST APIs, you must convert the standard AutoUpgrade configuration file format to JSON format.

   For example, here is a standard configuration file (`config.cfg`) for AutoUpgrade:

```
[oracle@localhost curl]$ more config.cfg
#Global configurations
#Autoupgrade's global directory, non-job logs generated,
#temp files created and other autoupgrade files will be
#send here
global.autoupg_log_dir=/autoupgrade/test/glog
#
# Database number 1
#
upg1.start_time=NOW
upg1.source_home=/databases/product/12.2.0/dbhome_1
upg1.target_home=/databases/product/19c/dbhome_1
upg1.sid=db122
upg1.log_dir=/autoupgrade/test/log
upg1.upgrade_node=localhost
upg1.target_version=19.1
```

   Here is the same configuration file, specifying the same parameter information, but rewritten as a JSON configuration input file (`config.json`) for use with the AutoUpgrade REST API method:

```
[oracle@localhost curl]$ more config.json
{
"global": {
        "autoupg_log_dir": "/autoupgrade/test/glog"
    },
    "jobs": [{
        "start_time": "NOW",
        "source_home": "/databases/product/12.2.0/dbhome_1",
        "target_home": "/databases/product/19c/dbhome_1",
        "sid": "db122",
        "log_dir": "/autoupgrade/test/log",
        "upgrade_node": "localhost",
        "target_version": "19.1"
    }]
}
```

How to Use AutoUpgrade REST API Authentication Privileges

To configure users with AutoUpgrade Administrator user privileges to access AutoUpgrade REST APIs, complete this setup procedure.

AutoUpgrade REST API has it's own authentication role for Oracle REST Data Services (ORDS), called `AutoUpgrade Administrator`. If the ORDS configuration parameter `autoupgrade.api.ordsauth` is set to `true`, then an ORDS user requires the `AutoUpgrade Administrator` role to use AutoUpgrade REST APIs.

To disable this feature, enter the following command:

```
java -jar ords.war set-property autoupgrade.api.ordsauth false
```

When `autoupgrade.api.ordsauth` is set to `true`, you must explicitly grant the `AutoUpgrade Administrator` role to users before they can access the AutoUpgrade REST APIs.

Before you enable or disable the AutoUpgrade REST API Administrator role, ensure that you have completed installation of the Oracle REST Data Services in standalone mode:

Configuring and Installing Oracle REST Data Services

1. After installing `ords.war` and before starting ORDS service, ensure that `autoupgrade.api.ordsauth` is set to true:

   ```
   java -jar ords.war set-property autoupgrade.api.ordsauth true
   ```

2. Create a user with the AutoUpgrade Administrator role. For example, the following command creates the user `roderigo` with the `AutoUpgrade Administrator` role:

   ```
   [oracle@localhost ords]$ java -jar ords.war user roderigo "AutoUpgrade
   Administrator"

   Enter a password for user roderigo:

   Confirm password for user roderigo:
   ```

3. Start the `ords.war` server.
4. Ensure that the user that you have created can access the AutoUpgrade endpoints successfully.

   The following is an example of successful authentication with user `roderigo`:

   ```
   [oracle@localhost curl]$ curl -u roderigo
   "http://localhost:8080/autoupgrade/tasks"

   Enter host password for user 'roderigo':


   {

       "total_tasks": 0,

       "tasks": [
   ```

```
    ]

}
```

The following is an example of unsuccessful authentication:

```
[oracle@localhost curl]$ curl "http://localhost:8080/autoupgrade/tasks"


{

    "code": "Unauthorized",

    "message": "Unauthorized",

    "type": "tag:oracle.com,2020:error/Unauthorized",

    "instance": "tag:oracle.com,2020:ecid/LI6fVLjXrI2rvmEFoBzAAw"

}
```

5. Create additional authenticated users as needed for the AutoUpgrade REST APIs.
6. Use the following command to disable the AutoUpgrade REST API authorization:

```
java -jar ords.war set-property autoupgrade.api.ordsauth false
```

7. Restart the ORDS server.
8. You should now be able to access AutoUpgrade REST API endpoints directly without requiring password authentication:

```
[oracle@localhost curl]$ curl "http://localhost:8080/autoupgrade/tasks"


{

    "total_tasks": 0,

    "tasks": [

    ]

}
```

REST APIs for AutoUpgrade

These REST APIs are available for the AutoUpgrade REST services feature in Oracle REST Data Services (ORDS).

- Create a New AutoUpgrade Task
  Submit an AutoUpgrade task through the REST API, either with or without a JSON configuration file.
- Progress Report Request for an AutoUpgrade Task
  Use the `progress` API to get a progress report of an AutoUpgrade task specified by the `taskid`.
- Console Request for an AutoUpgrade Task
  Use the `console` API to get a console for the AutoUpgrade task specified by the `taskid`.
- Task Details Request for AutoUpgrade
  Use the `task` API to get AutoUpgrade task details for the task specified by the `taskid`.
- Task List Request for AutoUpgrade Tasks
  Use the `tasks` API to get an AutoUpgrade task list.
- Log or Log List for an AutoUpgrade Task
  Use the `log` API to open a log, or to get a log list of a task specified by `taskid`.
- Status Report of Task Request for AutoUpgrade
  Use the `status` API to get a status report of an AutoUpgrade task specified by the `taskid`.

Create a New AutoUpgrade Task

Submit an AutoUpgrade task through the REST API, either with or without a JSON configuration file.

Method

POST

Path

```
/autoupgrade/task
```

Usage Notes

There are two uses:

1. Submit an AutoUpgrade task. To submit the task, the databases specified in the AutoUpgrade configuration file must be on the same host as the Oracle REST Data Services server instance, and the target database release must be a release that is supported for direct upgrade from the source database release. The AutoUpgrade configuration file must be uploaded through the HTTP or HTTPS request body in JSON format.
2. Resubmit an existing task through the AutoUpgrade REST APIs. When you resubmit a task by providing an existing `taskid` parameter, the configuration file previously associated with that task is used, so you do not need to provide a configuration file. However, you can specify different values for the parameters `mode` and `restore_on_fail`. For example: you can submit a task where the `mode` parameter is set to `analyze`. After the analyze tasks complete without any check failures, you can resubmit the task with the same `taskid`, but proceed to upgrade the database by changing the `mode` parameter to `deploy`.

Request

Path parameters:

Parameter	Description
`mode`	(Required) AutoUpgrade processing mode. Options: `analyze`, `fixups`, `deploy`, `upgrade`, `postfixups`.
`restore_on_fail`	(Optional) Generates a restore point during the upgrade. Options: `yes`, `no`
`taskid`	(Optional) Reruns the existing AutoUpgrade task that you specify with a `taskid`.

Response

- 201 Response: Description of the new AutoUpgrade task.
- 400 Response: Returned if parameters are missing.

Examples

Example A-1 Submit an AutoUpgrade Task Request

The following example shows how to submit an AutoUpgrade configuration file to create a new AutoUpgrade task in analyze mode by submitting a POST request on the REST resource using cURL

```
curl --data-binary "@config.json" -X POST --header "Content-Type:application/
json"
'http://localhost:8080/autoupgrade/task?mode=analyze'
```

In the JSON configuration file (`config.json`) for this analyze request, there is one source database instance, `db122`, which you specify to upgrade from Oracle Database 12c (12.2) to Oracle Database 19c:

```
{
"global": {
        "autoupg_log_dir": "/autoupgrade/test/glog"
    },
    "jobs": [{
        "start_time": "NOW",
        "source_home": "/databases/product/12.2.0/dbhome_1",
        "target_home": "/databases/product/19.0.0/dbhome_1",
        "sid": "db122",
        "log_dir": "/autoupgrade/test/log",
        "upgrade_node": "localhost",
        "target_version": "19.1"
    }]
}
```

The location of the returning header for this task request can be used to view the AutoUpgrade task.

```
Type:application/json" 'http://localhost:8080/autoupgrade/task?mode=analyze' -
i
HTTP/1.1 201 Created
Date: Thu, 14 Oct 2021 19:52:59 GMT
```

```
Set-Cookie: JSESSIONID=node01efwo2229s7kj11rxw2l2mltc10.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Location: http://localhost:8080/autoupgrade/task?
taskid=job_2021-10-14_15_53_00.085_0
Transfer-Encoding: chunked
```

The 201 response body for this analyze request shows the AutoUpgrade job status description, returned in JSON format:

```
{
    "taskid": "job_2021-10-14_15_53_00.085_0",
    "status": "submitted",
    "message": "",
    "link": "http://localhost:8080/autoupgrade/task?
taskid=job_2021-10-14_15_53_00.085_0",
    "config": {
        "global": {
            "autoupg_log_dir": "/autoupgrade/test/glog"
        },
        "jobs": [
            {
                "start_time": "NOW",
                "source_home": "/databases/product/12.2.0/dbhome_1",
                "target_home": "/databases/product/19.1.0/dbhome_1",
                "sid": "db122",
                "log_dir": "/autoupgrade/test/log",
                "upgrade_node": "localhost",
                "target_version": "19.1"
            }
        ]
    }
}
```

Example A-2 Response for an AutoUpgrade Job After a Faulty Request

In the following example, an AutoUpgrade job is submitted with an invalid `mode` parameter:

```
curl --data-binary "@config.json" -X POST --header "Content-Type:application/
json"
'http://localhost:8080/autoupgrade/task?mode=NULL'
```

The 400 response header to the faulty request is as follows:

```
Type:application/json" 'http://localhost:8080/autoupgrade/task?mode=null' -i
HTTP/1.1 400 Bad Request
Date: Wed, 20 Oct 2021 18:02:46 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
```

The following example shows this 400 response body, returned in JSON format:

```
{
    "taskid": "job_2021-10-20_14_02_46.855_4",
    "status": "Bad Request",
    "message": "AutoUpgrade parameter mode is invalid.",
    "link": "",
    "config": null
}
```

Example A-3 Rerunning an AutoUpgrade Job Using a `taskid`

In this example, Task ID `job_2021-10-14_15_53_00.085_0` is rerun in analyze mode:

```
curl -X POST 'http://localhost:8080/autoupgrade/task?
mode=analyze&taskid=job_2021-10-14_15_53_00.085_0'
```

The 201 response header for this rerun job is as follows:

```
HTTP/1.1 201 Created
Date: Mon, 10 Jan 2022 19:44:40 GMT
Set-Cookie: JSESSIONID=node0sjsoy3w4cntl13qnat979zc6n4.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Location: http://localhost:8080/autoupgrade/task?
taskid=job_2022-01-10_14_19_09.156_1
Transfer-Encoding: chunked
```

The task body describes the task in JSON format:

```
{    "taskid": "job_2021-10-14_15_53_00.085_0",    "status": "submitted",
"message": "",    "link": "http://localhost:8080/autoupgrade/task?
taskid=job_2021-10-14_15_53_00.085_0",
          "config": {
              "global": {
                  "autoupg_log_dir": "/autoupgrade/test/glog"
              },
              "jobs": [
                  {
                      "start_time": "NOW",
                      "source_home": "/databases/product/12.2.0/dbhome_1",
                      "target_home": "/databases/product/19.1.0/dbhome_1",
                      "sid": "db122",
                      "log_dir": "/autoupgrade/test/log",
                      "upgrade_node": "localhost",
                      "target_version": "19.1"
                  }
              ]
          } }
```

Progress Report Request for an AutoUpgrade Task

Use the `progress` API to get a progress report of an AutoUpgrade task specified by the `taskid`.

Method

GET

Path

```
/autoupgrade/progress
```

Usage Notes

When you submit the progress request for the task that you specify with the `taskid`, you receive a report of the progress of the AutoUpgrade task identified by that `taskid`.

Requests

Request parameters:

Parameter	Description
`taskid`	(Required) Identifies the task for which you want to obtain the progress report.

Response

- 200 Response: Reports the progress of the task that you specify with the `taskid`.
- 400 Response: Reports that the `taskid` is missing
- 404 Response: Reports that the `taskid` is invalid

Examples

Example A-4 Submit Task Progress Report GET Request

The following example shows how to get an AutoUpgrade task progress by providing an AutoUpgrade `taskid` with cURL.

```
curl http://localhost:8080/autoupgrade/progress?
taskid=job_2021-10-20_14_44_57.11_1
```

This is the 200 response to the preceding GET task progress report for a job:

```
HTTP/1.1 200 OK
Date: Wed, 20 Oct 2021 19:56:32 GMT
Set-Cookie: JSESSIONID=node01rkapzylgd2tljf28bghba9bx6.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
```

The 200 response body for this GET progress report for a job request is as follows, in JSON format:


```
{
    "taskid": "job_2021-10-20_14_44_57.11_1",
    "status": "successful",
    "message": "",
    "autoupgrade_progress": {
        "totalJobs": 1,
        "totalFinishedJobs": 1,
        "startTime": "2021-10-20 14:44:59",
        "lastUpdateTime": "2021-10-20 14:45:10",
        "jobs": [
            {
                "sid": "db122",
                "jobNo": 100,
                "isCDB": false,
                "totalPercentCompleted": 100,
                "totalContainers": 1,
                "totalStages": 1,
                "additionalInfo": "",
                "lastUpdateTime": "2021-10-20 14:45:10",
                "stages": [
                    {
                        "stage": "PRECHECKS",
                        "percentCompleted": "100",
                        "totalChecks": 104,
                        "totalCompletedChecks": 104,
                        "additionalInfo": "",
                        "lastUpdateTime": "2021-10-20 14:44:59",
                        "containers": [
                            {
                                "container": "db122",
                                "totalChecks": 104,
                                "completedChecks": 104,
                                "succeededChecks": 91,
                                "failedChecks": 13,
                                "runningChecks": [
                                ],
                                "finishedChecks": [
                                    "PLUGIN_COMP_COMPATIBILITY",
                                    "AMD_EXISTS",
                                    "COMPATIBLE_NOT_SET",
                                    "COMPATIBLE_PARAMETER",
                                    "CASE_INSENSITIVE_AUTH",
                                    "CYCLE_NUMBER",
                                    "CREATE_WINDOWS_SERVICE",
                                    "AUDTAB_ENC_TS",
                                    "CONC_RES_MGR",
                                    "DATA_MINING_OBJECT",
                                    "DV_ENABLED",
                                    "AWR_EXPIRED_SNAPSHOTS",
                                    "DV_SIMULATION",
                                    "DEPEND_USR_TABLES",
                                    "FLASH_RECOVERY_AREA_SETUP",
                                    "FILES_NEED_RECOVERY",
```

```
"INVALID_SYS_TABLEDATA",
"INVALID_OBJECTS_EXIST",
"JAVAVM_STATUS",
"ORACLE_RESERVED_USERS",
"HIDDEN_PARAMS",
"JVM_MITIGATION_PATCH",
"TRGOWNER_NO_ADMNDBTRG",
"MV_REFRESH",
"PENDING_DST_SESSION",
"RAISE_COMPATIBLE",
"COMPATIBLE_PFILES",
"INVALID_USR_TABLEDATA",
"POST_JVM_MITIGAT_PATCH",
"PURGE_RECYCLEBIN",
"PRE_FIXED_OBJECTS",
"SYNC_STANDBY_DB",
"SYS_DEFAULT_TABLESPACE",
"TWO_PC_TXN_EXIST",
"UNDERSCORE_EVENTS",
"UNIAUD_TAB",
"XBRL_VERSION",
"PARAMETER_DEPRECATED",
"XDB_RESOURCE_TYPE",
"PARAMETER_NEW_NAME_VAL",
"PARAMETER_OBSOLETE",
"PARAMETER_RENAME",
"MIN_ARCHIVE_DEST_SIZE",
"NEW_TIME_ZONES_EXIST",
"PARAMETER_MIN_VAL",
"DISK_SPACE_FOR_RECOVERY_AREA",
"PA_PROFILE",
"UPG_BY_STD_UPGRD",
"MIN_RECOVERY_AREA_SIZE",
"OLS_VERSION",
"ROLLBACK_SEGMENTS",
"ARCHIVE_MODE_ON",
"PGA_AGGREGATE_LIMIT",
"APEX_MANUAL_UPGRADE",
"TABLESPACES",
"OLAP_PAGE_POOL_SIZE",
"STANDARD_EDITION",
"TEMPTS_NOTEMPFILE",
"TS_FORUPG_STATUS",
"APPL_PRINCIPAL",
"TDE_IN_USE",
"STREAMS_SETUP",
"DIR_SYMLINKS",
"ORDIM_DESUPPORT",
"UTLRP_RUN_SERIAL",
"DESUPPORT_RAC_ON_SE",
"COMPATIBLE_GRP",
"EDS_EXISTS",
"UNIAUD_RECORDS_IN_FILE",
"JOB_TABLE_INTEGRITY",
"TARGET_CDB_COMPATIBILITY",
"DUPLICATE_DB_HASHES",
```

```
                                    "UNPLUG_PLUG_UPGRADE",
                                    "ENABLE_LOCAL_UNDO",
                                    "AUTO_LOGIN_KEYSTORE_REQUIRED",
                                    "NO_KEYSTORE_FILES",
                                    "INFORM_DROP_GRP",
                                    "PRE_DISABLE_BCT_UPG",
                                    "MAX_STRING_SIZE_ON_DB",
                                    "POST_RECOMPILE_IN_CDB",
                                    "TEMPTS_ALLOFFLINE",
                                    "PRE_RECOMPILE_IN_CDB",
                                    "MAX_DB_FILES_EXCEEDED",
                                    "POST_DISABLE_BCT_UPG",
                                    "TARGET_HOME_REGISTERED_INVENTORY",
                                    "ISRAC_SWITCHEDON_TARGETHOME",
                                    "KEYSTORE_CONFLICT",
                                    "WALLET_ROOT_KEYSTORE",
                                    "ORACLE_HOME_KEYSTORE",
                                    "DEPLOY_JOB_VALIDATIONS",
                                    "DUPLIC_SYS_SYSTEM_OBJS"
                                ],
                                "checksWithExecutionError": [
                                ],
                                "checksFailed": [
                                    "DICTIONARY_STATS",
                                    "POST_DICTIONARY",
                                    "POST_FIXED_OBJECTS",
                                    "OLD_TIME_ZONES_EXIST",
                                    "MANDATORY_UPGRADE_CHANGES",
                                    "RMAN_RECOVERY_VERSION",
                                    "TABLESPACES_INFO",
                                    "ORDIM_INFO_DESUPPORT",
                                    "POST_UTLRP",
                                    "COMPONENT_INFO",
                                    "INVALID_ORA_OBJ_INFO",
                                    "INVALID_APP_OBJ_INFO",
                                    "TIMESTAMP_MISMATCH"
                                ],
                                "additionalInfo": "",
                                "lastUpdateTime": "2021-10-20 14:45:10"
                            }
                        ]
                    }
                ],
                "summary": [
                    {
                        "stage": "PRECHECKS",
                        "complete": "Yes",
                        "duration": "11.08S"
                    }
                ]
            }
        ]
    }
}
```

Example A-5 Task Progress Report Request With an Invalid Task ID

In this example, the task report progress is submitted for an invalid task id:

```
curl http://localhost:8080/autoupgrade/progress?taskid=null
```

The 404 response header for the preceding invalid request is as follows:

```
HTTP/1.1 404 Not Found
Date: Wed, 20 Oct 2021 18:49:12 GMT
Set-Cookie: JSESSIONID=node06xkycdfv7k1f1bus3q43zefum2.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
```

The 404 response body for this request is as follows, returned in JSON format:

```
{
    "taskid": "null",
    "status": "Bad Request",
    "message": "taskid is invalid.",
    "link": "",
    "config": ""
}

}
```

Console Request for an AutoUpgrade Task

Use the `console` API to get a console for the AutoUpgrade task specified by the `taskid`.

Method

GET

Path

```
/autoupgrade/console
```

Usage Notes

Get console outputs of an AutoUpgrade task, which is specified by `taskid`.

Request

Path parameters:

Parameter	Description
`taskid`	(Required) Identifies the task for which you want to obtain the console outputs.

Response

- 200 Response: An AutoUpgrade console opens.
- 400 Response: Reports that the `taskid` is missing
- 404 Response: Reports that the `taskid` is invalid

Examples

Example A-6 Submit Task Console GET Request

The following example shows how to submit a console request by providing an AutoUpgrade `taskid` with cURL:

```
curl http://localhost:8080/autoupgrade/console?
taskid=job_2021-10-20_14_44_57.11_1
```

The 200 response header for the preceding GET task console request is as follows:

```
HTTP/1.1 200 OK
Date: Wed, 20 Oct 2021 20:09:04 GMT
Set-Cookie: JSESSIONID=node01hwo60q3c5irz1pnn4fa8hhm2k9.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: plain/text
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
```

The 200 response body for this request is as follows, reporting the progress of the AutoUpgrade job returned in plain (text) format:

```
AutoUpgrade tool launched with default options
Processing config file ...
+--------------------------------+
| Starting AutoUpgrade execution |
+--------------------------------+
1 databases will be analyzed
Job 100 database db122
Job 100 completed
------------------- Final Summary --------------------
Number of databases            [ 1 ]

Jobs finished                  [1]
Jobs failed                    [0]
Jobs restored                  [0]
Jobs pending                   [0]

Please check the summary report at:
/databases/working/job_2021-10-20_14_44_57.11_1/cfgtoollogs/upgrade/auto/
status/status.html
/databases/working/job_2021-10-20_14_44_57.11_1/cfgtoollogs/upgrade/auto/
status/status.log
```

Example A-7 Response Header for a Console Request With an Invalid Task ID

In this example, the console request is submitted for an invalid task id:

```
curl http://localhost:8080/autoupgrade/console?taskid=null
```

The preceding request returns the following 404 header response to the Console request:

```
HTTP/1.1 404 Not Found
Date: Wed, 20 Oct 2021 18:49:12 GMT
Set-Cookie: JSESSIONID=node06xkycdfv7k1f1bus3q43zefum2.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
```

The 404 response body for this Console request is as follows, returned in JSON format:

```
{
    "taskid": "null",
    "status": "Bad Request",
    "message": "taskid is invalid.",
    "link": "",
    "config": ""
}
```

Task Details Request for AutoUpgrade

Use the `task` API to get AutoUpgrade task details for the task specified by the `taskid`.

Method

GET

Path

```
/autoupgrade/task
```

Usage Notes

When you provide a `taskid`, you obtain details about the task that is associated with that `taskid`.

For example:

```
curl http://localhost:8080/autoupgrade/task?
taskid=job_2021-10-20_14_44_57.11_1
```

Request

Path parameters:

Parameter	Description
taskid	(Required) System-generated identifier for the task for which you want to obtain details

Response

- 200 Response: Reports details of the task that you specify with the `taskid`.
- 400 Response: Reports that the `taskid` is missing
- 404 Response: Reports that the `taskid` is invalid

Example A-8 Submit Task Details GET Request

The following example shows how to submit an AutoUpgrade get task details request by providing an AutoUpgrade `taskid` with cURL:

```
curl http://localhost:8080/autoupgrade/task?
taskid=job_2021-10-20_14_44_57.11_1
```

The 200 response to the preceding get task details request is as follows. The Location header returns the URI that you can use to view the AutoUpgrade task.

```
HTTP/1.1 200 OK
Date: Wed, 20 Oct 2021 18:45:20 GMT
Set-Cookie: JSESSIONID=node0tl6lz63xj3xd11i05p8bfmw8u1.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Location: http://localhost:8080/autoupgrade/task?
taskid=job_2021-10-20_14_44_57.11_1
Transfer-Encoding: chunked
```

The 200 response body to this GET task details request is as follows, returned in a JSON format:

```
{
    "taskid": " job_2021-10-20_14_44_57.11_1",
    "status": "submitted",
    "message": "",
    "link": "http://localhost:8080/autoupgrade/task?
taskid=job_2021-10-14_15_53_00.085_0",
    "config": {
        "global": {
            "autoupg_log_dir": "/autoupgrade/test/glog"
        },
        "jobs": [
            {
                "dbname": "db122",
                "start_time": "NOW",
                "source_home": "/databases/product/12.2.0/dbhome_1",
                "target_home": "/databases/product/19c/dbhome_1",
                "sid": "db122",
                "log_dir": "/autoupgrade/test/log",
```

```
                "upgrade_node": "localhost",
                "target_version": "19.1"
            }
        ]
    }
}
```

Example A-9 Task Details Request with an Invalid `taskid`

A GET request with an invalid `taskid` is submitted:

```
curl http://localhost:8080/autoupgrade/task?taskid=null
```

The 404 response header for the invalid request is as follows:

```
{HTTP/1.1 404 Not Found
Date: Wed, 20 Oct 2021 18:49:12 GMT
Set-Cookie: JSESSIONID=node06xkycdfv7k1f1bus3q43zefum2.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
```

The 404 response body to this request is as follows, returned in JSON format:

```
{
    "taskid": "null",
    "status": "Bad Request",
    "message": "taskid is invalid.",
    "link": "",
    "config": ""
}
```

Task List Request for AutoUpgrade Tasks

Use the `tasks` API to get an AutoUpgrade task list.

Method

GET

Path

```
/autoupgrade/tasks
```

Requests

There are no parameters.

Response

200 Response: Reports details of all tasks.

Usage Notes

Use this method to obtain a list of tasks.

Example A-10 Submit Task List GET Request

The following example shows how to get an AutoUpgrade task list with cURL:

```
curl http://localhost:8080/autoupgrade/tasks
```

The following example shows a response header for a GET task list request:

```
HTTP/1.1 200 OK
Date: Wed, 20 Oct 2021 19:51:46 GMT
Set-Cookie: JSESSIONID=node0yawyihk3msz91eeauxijfaw6z3.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
```

The 200 response body for this list request is as follows, returned in JSON format:

```
{
    "total_tasks": 2,
    "tasks": [
        {
            "mode": "analyze",
            "taskid": "job_2021-10-20_14_44_57.11_1",
            "config": {
                "jobs": [
                    {
                        "source_home": "/databases/product/12.2.0/dbhome_1",
                        "sid": "db122"
                    }
                ]
            },
            "link": "http://localhost:8080/autoupgrade/task?
taskid=job_2021-10-20_14_44_57.11_1"
        },
        {
            "mode": "analyze",
            "taskid": "job_2021-10-20_15_54_01.568_2",
            "config": {
                "jobs": [
                    {
                        "source_home": "/databases/product/12.2.0/dbhome_1",
                        "sid": "db122"
                    }
                ]
            },
            "link": "http://localhost:8080/autoupgrade/task?
taskid=job_2021-10-20_15_54_01.568_2"
        }
```

```
    ]
}
```

Log or Log List for an AutoUpgrade Task

Use the `log` API to open a log, or to get a log list of a task specified by `taskid`.

Method

GET

Path

`/autoupgrade/log`

Usage Notes

When you submit a log request with the type or name, you receive an AutoUpgrade log for the task, specified by `taskid`. If you do not provide the type or name, then you receive an AutoUpgrade log list, which includes all HTML, JSON, and XML log files for the task, specified by `taskid`.

Request

Request parameters:

Parameter	Description
`taskid`	(Required) Specifies the AutoUpgrade task ID for which you want to open a log, or list logs.
`type`	(Optional) Specifies the type of an AutoUpgrade log. Options: `autolog`: AutoUpgrade log file `errlog`: Errors log file `userlog`: user log file
`name`	(Optional) Specify the filename of a specific log within a task ID log list.

Response

- 200 Response: AutoUpgrade opens the AutoUpgrade task err/user/auto log for the `taskid` that you specify.
- 400 Response: Reports that the `taskid` is missing
- 404 Response: Reports that the `taskid` is invalid

Examples

Example A-11 Submit Task Log GET Request

The following example shows how to get an AutoUpgrade user log by providing an AutoUpgrade `taskid` and `type` with cURL:

```
curl "http://localhost:8080/autoupgrade/log?
type=userlog&taskid=job_2021-10-20_14_44_57.11_1"
```

The 200 response header to the preceding task log request with `type` specified as `userlog` is as follows:

```
HTTP/1.1 200 OK
Date: Wed, 20 Oct 2021 20:30:02 GMT
Set-Cookie: JSESSIONID=node0jrq0bg40ya86mr8icb87d8z16.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: plain/text
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
```

The 200 response body for this request is as follows, returned in plain (text) format for a task log with `type` specified as `userlog`:

```
2021-10-20 14:44:57.737 INFO
build.hash:71778e3
build.version:23.0.0
build.date:2021/05/05 10:42:40 -0700
build.max_target_version:23
build.supported_target_versions:12.2,18,19,21,23
build.type:beta
build.hash_date:2021/04/15 15:06:03 -0700
build.label:rest_api

2021-10-20 14:44:57.791 INFO Loading user config file metadata
2021-10-20 14:44:58.598 INFO The target_version parameter was updated from
19.1 to 19.3.0.0.0 due to finding a more accurate value
2021-10-20 14:44:58.844 INFO srvctl command is not available or user does not
have permission to execute it
2021-10-20 14:44:59.163 INFO Finished processing dbEntry job_0
2021-10-20 14:44:59.475 INFO Current settings Initialized
2021-10-20 14:44:59.576 INFO Job 100 database db122
2021-10-20 14:45:11.685 INFO Job 100 completed
```

Example A-12 Response Header for a Log Request With an Invalid Task ID

In this example, the GET log request is submitted for an invalid task id:

```
curl http://localhost:8080/autoupgrade/log?taskid=null
```

This request returns the following 404 header response:

```
HTTP/1.1 404 Not Found
Date: Wed, 20 Oct 2021 18:49:12 GMT
Set-Cookie: JSESSIONID=node06xkycdfv7k1f1bus3q43zefum2.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
```

The 404 response body for the invalid `taskid` is as follows, returned in JSON format:

```
{
    "taskid": "null",
    "status": "Bad Request",
    "message": "taskid is invalid.",
    "link": "",
    "config": ""
}
```

Example A-13 Response Body for a Task ID Log List

The following example shows how to get an AutoUpgrade log list by providing an AutoUpgrade `taskid` with cURL. :

```
 curl 'http://localhost:8080/autoupgrade/log?
taskid='job_2022-01-10_14:19:09.156_1
```

The following example shows the 200 response body showing the log list for the `taskid`, returned in JSON format:

```
{
    "logs": [
        {
            "filename": "cfgtoollogs/upgrade/auto/autoupgrade.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=cfgtoollogs/upgrade/auto/
autoupgrade.log"
        },
        {
            "filename": "cfgtoollogs/upgrade/auto/autoupgrade_user.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=cfgtoollogs/upgrade/auto/
autoupgrade_user.log"
        },
        {
            "filename": "cfgtoollogs/upgrade/auto/autoupgrade_err.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=cfgtoollogs/upgrade/auto/
autoupgrade_err.log"
        },
        {
            "filename": "cfgtoollogs/upgrade/auto/status/status.json",
            "link": "http://localhost:8080/autoupgrade/log?
```

```
taskid=job_2022-01-10_14_19_09.156_1&name=cfgtoollogs/upgrade/auto/status/
status.json"
        },
        {
            "filename": "cfgtoollogs/upgrade/auto/status/progress.json",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=cfgtoollogs/upgrade/auto/status/
progress.json"
        },
        {
            "filename": "cfgtoollogs/upgrade/auto/status/status.html",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=cfgtoollogs/upgrade/auto/status/
status.html"
        },
        {
            "filename": "cfgtoollogs/upgrade/auto/status/status.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=cfgtoollogs/upgrade/auto/status/
status.log"
        },
        {
            "filename": "db122/100/autoupgrade_20220110.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/100/autoupgrade_20220110.log"
        },
        {
            "filename": "db122/100/autoupgrade_20220110_user.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/100/
autoupgrade_20220110_user.log"
        },
        {
            "filename": "db122/100/autoupgrade_err.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/100/autoupgrade_err.log"
        },
        {
            "filename": "db122/100/prechecks/prechecks_db122.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/100/prechecks/
prechecks_db122.log"
        },
        {
            "filename": "db122/100/prechecks/db122_checklist.xml",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/100/prechecks/
db122_checklist.xml"
        },
        {
            "filename": "db122/100/prechecks/db122_checklist.json",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/100/prechecks/
db122_checklist.json"
        },
        {
```

```
            "filename": "db122/100/prechecks/db122_preupgrade.html",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/100/prechecks/
db122_preupgrade.html"
        },
        {
            "filename": "db122/100/prechecks/upgrade.xml",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/100/prechecks/upgrade.xml"
        },
        {
            "filename": "db122/100/prechecks/db122_preupgrade.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/100/prechecks/
db122_preupgrade.log"
        },
        {
            "filename": "db122/101/autoupgrade_20220110.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/101/autoupgrade_20220110.log"
        },
        {
            "filename": "db122/101/autoupgrade_20220110_user.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/101/
autoupgrade_20220110_user.log"
        },
        {
            "filename": "db122/101/autoupgrade_err.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/101/autoupgrade_err.log"
        },
        {
            "filename": "db122/101/prechecks/prechecks_db122.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/101/prechecks/
prechecks_db122.log"
        },
        {
            "filename": "db122/101/prechecks/db122_checklist.xml",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/101/prechecks/
db122_checklist.xml"
        },
        {
            "filename": "db122/101/prechecks/db122_checklist.json",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/101/prechecks/
db122_checklist.json"
        },
        {
            "filename": "db122/101/prechecks/db122_preupgrade.html",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/101/prechecks/
db122_preupgrade.html"
        },
```

```
        {
            "filename": "db122/101/prechecks/upgrade.xml",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/101/prechecks/upgrade.xml"
        },
        {
            "filename": "db122/101/prechecks/db122_preupgrade.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/101/prechecks/
db122_preupgrade.log"
        },
        {
            "filename": "db122/102/autoupgrade_20220110.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/102/autoupgrade_20220110.log"
        },
        {
            "filename": "db122/102/autoupgrade_20220110_user.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/102/
autoupgrade_20220110_user.log"
        },
        {
            "filename": "db122/102/autoupgrade_err.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/102/autoupgrade_err.log"
        },
        {
            "filename": "db122/102/prechecks/prechecks_db122.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/102/prechecks/
prechecks_db122.log"
        },
        {
            "filename": "db122/102/prechecks/db122_checklist.xml",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/102/prechecks/
db122_checklist.xml"
        },
        {
            "filename": "db122/102/prechecks/db122_checklist.json",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/102/prechecks/
db122_checklist.json"
        },
        {
            "filename": "db122/102/prechecks/db122_preupgrade.html",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/102/prechecks/
db122_preupgrade.html"
        },
        {
            "filename": "db122/102/prechecks/upgrade.xml",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/102/prechecks/upgrade.xml"
        },
```

```
        {
            "filename": "db122/102/prechecks/db122_preupgrade.log",
            "link": "http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/102/prechecks/
db122_preupgrade.log"
        }
    ]
}
```

Example A-14 Request a Specific Log with `name` Parameter

The following example shows how to request a specific log file within a task ID log list, by providing the filename with the `name` parameter, and the `taskid` of the task that generated the log:

```
curl 'http://localhost:8080/autoupgrade/log?
taskid=job_2022-01-10_14_19_09.156_1&name=db122/102/
autoupgrade_20220110_user.log'
```

This request returns this 200 response body that displays the log file information for the filename specified:

```
2022-01-10 14:44:43.447 INFO
build.hash:71778e3
build.version:23.0.0
build.date:2021/05/05 10:42:40 -0700
build.max_target_version:23
build.supported_target_versions:12.2,18,19,21,23
build.type:beta
build.hash_date:2021/04/15 15:06:03 -0700
build.label:rest_api

2022-01-10 14:44:49.529 INFO Analyzing db122, 104 checks will run using 4
threads
```

Status Report of Task Request for AutoUpgrade

Use the `status` API to get a status report of an AutoUpgrade task specified by the `taskid`.

Method

GET

Path

```
/autoupgrade/status
```

Usage Notes

To receive an AutoUpgrade task status in JSON format, submit an AutoUpgrade `taskid`.

Request

Path parameters:

Parameter	Description
`taskid`	(Required) System-generated identifier for the task for which you want to obtain status.

Response

- 200 Response: Reports the status of the task that you specify with the `taskid`.
- 400 Response: Reports that the `taskid` is missing
- 404 Response: Reports that the `taskid` is invalid

Examples

Example A-15 Submit Task Status GET Request

The following example shows how to get an AutoUpgrade task status request with cURL:

```
curl http://localhost:8080/autoupgrade/status?
taskid=job_2021-10-20_14_44_57.11_1
```

This is the 200 response header for the preceding GET task status report request:

```
HTTP/1.1 200 OK
Date: Wed, 20 Oct 2021 19:51:46 GMT
Set-Cookie: JSESSIONID=node0yawyihk3msz91eeauxijfaw6z3.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
```

This is the response body for the GET task status report request, in JSON format:

```
{
    "taskid": "job_2021-10-20_14_44_57.11_1",
    "status": "successful",
    "message": "",
    "autoupgrade_status": {
        "totalJobs": 1,
        "lastUpdateTime": "2021-10-20 14:45:11",
        "jobs": [
            {
                "sid": "db122",
                "jobNo": 100,
                "logDirectory": "/databases/working/
job_2021-10-20_14_44_57.11_1/db122/100",
                "conNumber": 1,
                "lastUpdateTime": "2021-10-20 14:45:11",
                "modules": [
                    {
                        "moduleName": "PRECHECKS",
```

```
                        "status": 0,
                        "errors": [
                        ],
                        "lastUpdateTime": "2021-10-20 14:45:11"
                    }
                ]
            }
        ]
    }
}
```

Example A-16 Submit Task Status GET Request with an invalid `taskid`

In the following example, an AutoUpgrade GET task status request is submitted with an invalid `taskid` parameter:

```
curl http://localhost:8080/autoupgrade/status?taskid=null
```

The 404 response header for the preceding invalid GET task status request is as follows:

```
HTTP/1.1 404 Not Found
Date: Wed, 20 Oct 2021 18:49:12 GMT
Set-Cookie: JSESSIONID=node06xkycdfv7k1f1bus3q43zefum2.node0; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked
```

The response body for this invalid GET task status request is as follows, in JSON format:

```
{
    "taskid": "null",
    "status": "Bad Request",
    "message": "taskid is invalid.",
    "link": "",
    "config": ""
}
```

B

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.
- Pre-Upgrade Information Tool and AutoUpgrade Preupgrade
  Learn how to obtain the same features previously offered through the Pre-Upgrade information tool (`preupgrade.jar`) by using AutoUpgrade with the `preupgrade` clause.

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 B-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`). Run these scripts in UPGRADE mode. 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.
`dbupgrade`, `dbupgrade.cmd`	Shell scripts that call the `catctl.pl` script. 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.

Table B-1 (Cont.) Upgrade, Post-Upgrade, and Downgrade Scripts

Script	Description
catdwgrd.sql	This is the downgrade script, which is used in the procedure to downgrade to the earlier release from which you upgraded.
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.
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 .
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.
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.
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.

Table B-1 (Cont.) Upgrade, Post-Upgrade, and Downgrade Scripts

Script	Description
`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`.

Pre-Upgrade Information Tool and AutoUpgrade Preupgrade

Learn how to obtain the same features previously offered through the Pre-Upgrade information tool (`preupgrade.jar`) by using AutoUpgrade with the `preupgrade` clause.

Starting with Oracle Database 21c, the Pre-Upgrade Information Tool is no longer offered. All features that you previously obtained using the Pre-Upgrade Information Tool are now offered using the AutoUpgrade tool

- Using AutoUpgrade To Obtain Pre-Upgrade Information Tool Checks
  Learn how to obtain the same system checks you obtained previously through the Pre-Upgrade Information Tool (`preupgrade.jar`) by using the AutoUpgrade utility.
- Examples of Preupgrade and Postupgrade Checks
  Review the upgrade scenario examples to understand the differences between the Pre-Upgrade Information Tool and AutoUpgrade using the `preupgrade` parameter.

Using AutoUpgrade To Obtain Pre-Upgrade Information Tool Checks

Learn how to obtain the same system checks you obtained previously through the Pre-Upgrade Information Tool (`preupgrade.jar`) by using the AutoUpgrade utility.

To see how to use AutoUpgade to perform a check you performed in the past using the Pre-Upgrade Information tool, review the comparison table that follows, and review the examples in this topic.

For full details about how to use the AutoUpgrade utility to perform checks and run scripts, refer to the topic Preupgrade under "AutoUpgrade Command Line Parameters."

Table B-2 Comparison of Preupgrade and Postupgrade Checks

Check	Pre-Upgrade Information Tool	AutoUpgrade
Check source Oracle Database release readiness for upgrade	Run `preupgrade.jar` for database checks. The generated file `upgrade.xml` shows check results.	Run AutoUpgrade with the `preupgrade` parameter, run in `analyze` mode. Obtain results from `upgrade.xml`, which is located in the directory *`Log_dir`*`/`*`database_name`*`/`*`job_number`*`/prechecks/`.

Table B-2 (Cont.) Comparison of Preupgrade and Postupgrade Checks

Check	Pre-Upgrade Information Tool	AutoUpgrade
Obtain and run scripts to fix some issues found in the source Oracle Database to prepare for upgrade.	Run `preupgrade.jar` to obtain scripts (`preupgrade_fixups.sql` and `preupgrade_fixups_pdbname.sql`), which you run manually using SQL*Plus	Run Autoupgrade with the `preupgrade` parameter, run in `fixups` mode. AutoUpgrade runs all database checks, and on the basis of those results, runs fixups automatically.
Obtain and run scripts to perform some postupgrade fixes on the upgraded Oracle Database	Run `preupgrade.jar` to obtain scripts (`postupgrade_fixups.sql` and `postupgrade_fixups_pdbname.sql`), which you run manually using SQL*Plus	Run Autoupgrade with the `preupgrade` parameter, run in `postfixups` mode. AutoUpgrade runs all database checks, and on the basis of those results, runs postfixups automatically.

Related Topics

- 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.

Examples of Preupgrade and Postupgrade Checks

Review the upgrade scenario examples to understand the differences between the Pre-Upgrade Information Tool and AutoUpgrade using the `preupgrade` parameter.

In the examples that follow, consider the following configurations:

- Source: Oracle Database 12c Release 2 (12.2.0.1) Enterprise Edition
- Oracle home: `/u01/app/oracle/product/12.2.0/dbhome_1`
- Oracle System Identifier (SID): `db122`
- Target: Oracle Database 19c for the Pre-Upgrade Information Tool, and Oracle Database 21c for AutoUpgrade using the `preupgrade` parameter
- Target release Oracle home for Pre-Upgrade Information Tool: `/u01/app/oracle/product/19.0.0/dbhome_1`
- Target release Oracle home for AutoUpgrade using the `preupgrade` parameter: `/u01/app/oracle/product/21.0.0/dbhome_1`

Example B-1 Non-CDB Upgrade Database Checks Using AutoUpgrade Preupgrade Parameter

Previously, when you used the Pre-Upgrade Information Tool, to run all database prechecks, you completed steps in this order:

1. Run `setenv ORACLE_HOME /u01/app/oracle/product/12.2.0/dbhome_1`
2. Run `setenv ORACLE_SID db122`
3. Switch directory to the target Oracle home: `/u01/app/oracle/product/19.0.0/dbhome_1/rdbms/admin`
4. Run the Pre-Upgrade Information Tool:, directing output to a directory: `java -jar preupgrade.jar dir /user/log`

5. Check the `upgrade.xml` file in the log directory
6. Find `preupgrade_fixups.sql` and `postupgrade_fixups.sql` in the log directory you specified (`/user/log/cfgtoollogs/db122/preupgrade`).

Now, with the AutoUpgrade tool using the `preupgrade` parameter, you run steps in this order:

1. Run `setenv ORACLE_HOME /u01/app/oracle/product/12.2.0/dbhome_1`
2. Run `setenv ORACLE_SID db122`
3. Run `java -jar autoupgrade.jar -preupgrade "target_version=21,dir=/autoupgrade/test/log"`
4. Check `upgrade.xml` under the directory `/autoupgrade/test/log/db122/100/prechecks`

Example B-2 Non-CDB Upgrade Running Prefixups Using AutoUpgrade Preupgrade Parameter

Previously, when you used the Pre-Upgrade Information Tool, to run prefixups on your source database, you completed steps in this order:

1. Change directory to the source Oracle home: `/u01/app/oracle/product/12.2.0/dbhome_1`
2. Start SQL*Plus
3. Using SQL*Plus, run `preupgrade_fixups.sql` generated in the directory you specified (`/user/log/cfgtoollogs/db122/preupgrade`)
4. Check fixups results from the SQL*Plus output.

Now, with the AutoUpgrade tool using the `preupgrade` parameter, you run preupgrade fixups as follows:

1. Run `setenv ORACLE_HOME /u01/app/oracle/product/12.2.0/dbhome_1`.
2. Run `setenv ORACLE_SID db122`.
3. Run `java -jar autoupgrade.jar -preupgrade "target_version=21,dir=/autoupgrade/test/log" -mode fixups`. The scripts run automatically.
4. Check `prefixups.xml` under the directory `/autoupgrade/test/log/db122/102/prefixups`.

Example B-3 Non-CDB Upgrade Running Postupgrade Fixups Using AutoUpgrade Preupgrade Parameter

Previously, when you used the Pre-Upgrade Information Tool, to run postupgrade fixups on your source database, you completed steps in this order:

1. Start SQL*Plus under the source Oracle Home: `/u01/app/oracle/product/12.2.0/dbhome_1`
2. Using SQL*Plus, run `postupgrade_fixups.sql` generated in the directory you specified (`/user/logcfgtoollogs/db122/preupgrade`).
3. Check fixups results from SQL*Plus output.

Now, with the AutoUpgrade tool using the `preupgrade` parameter, you run postupgrade fixups as follows:

1. Run `setenv ORACLE_HOME /u01/app/oracle/product/12.2.0/dbhome_1`.
2. Run `setenv ORACLE_SID db122`.

3. Run `java -jar autoupgrade.jar -preupgrade "target_home=/u01/app/oracle/product/21.0.0/dbhome_1,dir=/autoupgrade/test/log" -mode postfixups`
4. Check `postfixups.xml` under directory `/autoupgrade/test/log/db122/102/postfixups`.

Example B-4 Comparison of Output Examples for Pre-Upgrade Information Tool and AutoUpgrade

The following is an example of the output generated by running prechecks with the Pre-Upgrade Information tool (using Oracle Database 19c as the target release), and AutoUpgrade using the Pre-Upgrade Information Tool (using an upgrade to Oracle Database 21c). Note that you are required to run scripts manually after the script completes:

```
[oracle@localhost preupgrade]$ java -jar /databases/product/19c/dbhome_1/
rdbms/admin/preupgrade.jar \
xml dir /autoupgrade/test/preupgrade
==================
PREUPGRADE SUMMARY
==================
 /autoupgrade/test/preupgrade/upgrade.xml
 /autoupgrade/test/preupgrade/preupgrade_fixups.sql
 /autoupgrade/test/preupgrade/postupgrade_fixups.sql
Execute fixup scripts as indicated below:
Before upgrade:
Log into the database and execute the preupgrade fixups
@/autoupgrade/test/preupgrade/preupgrade_fixups.sql
After the upgrade:
Log into the database and execute the postupgrade fixups
@/autoupgrade/test/preupgrade/postupgrade_fixups.sql
Preupgrade complete: 2020-07-06T16:35:14
```

The following is an example for running AutoUpgrade with the `preupgrade` parameter using `analyze` mode, with the target release Oracle Database 21c.

```
[oracle@localhost /autoupgrade]$ java -jar autoupgrade.jar -preupgrade
"target_version=21,dir=/autoupgrade/test/log" -mode analyze
AutoUpgrade tool launched with default options
Processing config file ...
+--------------------------------+
| Starting AutoUpgrade execution |
+--------------------------------+
1 databases will be analyzed
Type 'help' to list console commands
upg> Job 102 completed
------------------- Final Summary --------------------
Number of databases [ 1 ]
Jobs finished successfully [1]
Jobs failed [0]
Jobs pending [0]
------------- JOBS FINISHED SUCCESSFULLY -------------
Job 102 for db122
```

C

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.

- Requirements for Using DBUA
  To use Database Upgrade Assistant (DBUA) for multitenant architecture and non-CDB Oracle Database upgrades, complete these requirements.
- 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.
- 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.
- Moving a Database from an Existing Oracle Home
  You can use Database Upgrade Assistant (DBUA) to migrate Oracle Database databases from an existing Oracle home to another 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.
- 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.
- Migrating from Oracle ACFS to Alternative Storage Before Using DBUA
  If your operating system is Microsoft Windows, and your Oracle Database release uses Oracle Advanced Cluster File System (ACFS), then you must migrate from that storage before starting the upgrade.

Requirements for Using DBUA

To use Database Upgrade Assistant (DBUA) for multitenant architecture and non-CDB Oracle Database upgrades, complete these requirements.

DBUA and Databases Using Oracle Database Vault

If Oracle Database Vault is enabled, then review the topic “Requirement for Upgrading Oracle Databases That Use Oracle Database Vault”.

DBUA and Oracle Database Architecture Configuration

Oracle Database 19c is the terminal release for non-CDB architecture.

> **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.

You can use DBUA to upgrade multitenant architecture container databases (CDB), pluggable databases (PDBs), and non-CDB databases. The procedures are the same, but the choices you must make and the behavior of DBUA are different, depending on the type of upgrade:

- 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 (Non-CDB single instance, 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.

Downloads Required Before Using DBUA

Before you run DBUA, complete the following downloads:

- So that DBUA can perform preupgrade checks, you must download and install the most recent AutoUpgrade utility from My Oracle Support note 2485457.1.
- For upgrades from Oracle Database 19c, download the latest version of `preupgrade.jar` from My Oracle Support note 884522.1. Download `preupgrade.jar` into the Oracle Database 19c Oracle home, under the directory path *`Oracle_home`*`/rdbms/admin`, and unzip the file there.

Related Topics

- 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.
- My Oracle Support note 2485457.1
- My Oracle Support note 884522.1

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 AutoUpgrade using the preupgrade parameter (`autoupgrade.jar -preupgrade -mode`), which runs database checks and preupgrade fixups that fix most issues before you start an upgrade.

DBUA 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.

Related Topics

- 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.

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 21c:

     ```
     ./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. 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 button 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**. Select a path where you want to place the backup.

     – **Create a New Partial Offline RMAN Backup with R/O User Tablespace**. If you select this option, then user tablespaces are placed into read-only mode during the upgrade, and a new partial offline RMAN backup is created.

     – **Use Latest Available Full RMAN Backup.** If you select this option, then click **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, 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.

Moving a Database from an Existing Oracle Home

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.

   After you complete you review, 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 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 C-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 C-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 C-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*

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
  ```

Migrating from Oracle ACFS to Alternative Storage Before Using DBUA

If your operating system is Microsoft Windows, and your Oracle Database release uses Oracle Advanced Cluster File System (ACFS), then you must migrate from that storage before starting the upgrade.

Starting with Oracle Database 21c, the Oracle Grid Infrastructure feature Oracle Advanced Cluster File System (Oracle ACFS) is desupported with Microsoft Windows. Accordingly, if your source Oracle Database release used Oracle ACFS for storage, then you must migrate to other storage before starting the upgrade. If your source Oracle Database release uses Oracle ACFS for storage when the upgrade is started, then you receive the error "PRVH-0570 : Oracle ACFS was found configured with resources," specifying resources that use the desupported storage feature.

Starting with Oracle Database 21c, the name of Oracle Automatic Storage Management Cluster File System (Oracle ACFS) is changed to Oracle Advanced Cluster File System (Oracle ACFS).

For Oracle Real Application Clusters files, in place of Oracle ACFS, Oracle recommends that you use Oracle ASM. For generic files, depending on your use case, Oracle recommends that you either move files to Oracle Database File System (DBFS), or move files to Microsoft Windows shared files.

Before you restart the upgrade, ensure that the following is true:

- Oracle ACFS is no longer in use.
- No Oracle ACFS resources, such as volumes, or file system resources, are present in the current configuration.
- All Oracle ACFS resources indicated in the PRVH-0570 error messasge are deconfigured and removed.

> **Caution:**
>
> After upgrade to Oracle Database 21c, no data can be recovered from files stored on Oracle ACFS, because Oracle ACFS is not available in Oracle Database 21c and later releases.

For more information about this requirement, refer to My Oracle Support note 1369107.1.

Related Topics

- My Oracle Support Note 1369107.1 ACFS Support On OS Platforms (Certification Matrix)

D

AutoUpgrade Error Messages

AutoUpgrade provides a set of errors and messages for failures in preupgrade, upgrade, and postupgrade errors.

Host Errors (UPG-1000 to UPG-1200)

- UPG-1000: 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
- UPG-1001: There was a problem reading the state file perhaps there was corruption writing the file and in the next write it might be fixed
- UPG-1002: Error deserializing the object for rerun, review log for any errors
- UPG-1100: Either the console or the job manager were interrupted and it might lead to unknown errors because if there were active tasks running they won't be able to be finish gracefully nor we will guarantee that we track their progress properly
- UPG-1200: Error reading the standard input, perhaps a malformed character or there was a problem with the process's stdin

Preupgrade Errors (UPG-1300 to UPG-1319

- UPG-1300: The current execution of the database fixups was interrupted and a set of checks is still pending
- UPG-1301: The current execution of the database fixups was not completed
- UPG-1302: The current execution of the database checks was interrupted
- UPG-1303: A failed check has an ERROR severity but the fixup is unavailable or failed to correct the problem. Manually fix the problem and rerun AutoUpgrade
- UPG-1304: IO error creating before, during or after upgrade pfile
- UPG-1305: Error executing SQL statement in upgrade inspector during database preupgrade checks
- UPG-1306: Error during the database preupgrade checks in upgrade inspector
- UPG-1307: Error running database postupgrade checks in upgrade inspector
- UPG-1308: Error running database postupgrade checks in upgrade inspector
- UPG-1309: Error accessing the log file that contains the check list (CFG format)
- UPG-1310: Error accessing the log file that contains the check list (log format)
- UPG-1311: Error accessing the checks HTML report file
- UPG-1312: A failed check has an ERROR severity but the fixup is unavailable or failed to correct the problem. Manually fix the problem and rerun AutoUpgrade
- UPG-1313: Internal error, contact Oracle Support
- UPG-1314: User requested an abort of the checks
- UPG-1315: User requested an abort of the fixups

- UPG-1316: Error running database checks or fixups
- UPG-1317: Required checklist file unavailable; unable to run fixups
- UPG-1318: Error creating custom pfiles
- UPG-1319: Loading the current state of the database failed

Upgrade Process Errors

- UPG-1400: Database upgrade failed with errors
- UPG-1401: Opening database for upgrade in the target home failed
- UPG-1402: GRP creation prior to upgrade failed
- UPG-1403: Drop of existing GRP failed
- UPG-1404: Error running upgrade
- UPG-1405: Database upgrade was interrupted
- UPG-1406: Unexpected Exception occurred during database compilations
- UPG-1407: IO error creating database spfile
- UPG-1408: Unable to open pdb in upgrade mode
- UPG-1409: Unable to open pdb in normal mode
- UPG-1410: Unable to open database in normal mode
- UPG-1411: Invalid version of catctl.pl and catctl.pm
- UPG-1412: Unable to query database
- UPG-1413: Error occurred during upgrade
- UPG-1414: Database compilation interrupted
- UPG-1415: Database upgrade interrupted by a system shutdown or CTRL+C
- UPG-1416: Error during database compilations
- UPG-1417: PDB upgrade failed with errors
- UPG-1418: Database upgrade failed; review the upgrade log files
- UPG-1419 Database upgrade job has been killed, by abort or restore
- UPG-1420 Enabling Local Undo failed

Postupgrade Errors

- UPG-1500: Database post upgrade failed
- UPG-1501: Oratab file not readable
- UPG-1502: Oratab file not writable
- UPG-1503: IO error updating the oratab file
- UPG-1504: Error writing to oratab file
- UPG-1505: Error copying listener.ora file in postupgrade
- UPG-1506: Error copying tnsnames.ora file in postupgrade
- UPG-1507: Error copying sqlnet.ora file in postupgrade
- UPG-1508: Error processing the ifile
- UPG-1509: Error copying the wallet files

- UPG-1510: Unable to create the postupgrade directory
- UPG-1511: Error during user-defined postupgrade action
- UPG-1512: Error copying the password files
- UPG-1513: Error restoring the database state
- UPG-1514: Error deserializing the database state
- UPG-1515: Error dropping the GRP
- UPG-1516: Error restarting the databas
- UPG-1517: Error recreating the final spfile
- UPG-1518: An error occurred while copying Context text files in Postupgrade phase
- UPG-1519: An error occurred while generating Context text file list in Drain phase
- UPG-1520: Error copying oranfstab file in postupgrade
- UPG-1521: Error copying ldap.ora file in postupgrade
- UPG-1522: Error updating network file's ORACLE_HOME in postupgrade
- UPG-1523: Error restarting the pdbs

Database Upgrade Stage Errors

- UPG-1600: Error copying the pfile during an UPGRADE mode job
- UPG-1601: Error reading the error properties file
- UPG-1602: Job killed
- UPG-1603: Error executing a database command for PDB$SEED
- UPG-1604: Error executing a database command
- UPG-1605: Unable to abort a job in a unsupported stage
- UPG-1606: Source version cannot be upgraded to the specified target version
- UPG-1607: Target version is not supported by AutoUpgrade
- UPG-1608: Unable to determine database version; review the logs files
- UPG-1699: Error finding error definition, contact Oracle Support
- UPG-1700: Error generating Windows services in drain module
- UPG-1701: Error stopping queue jobs during drain
- UPG-1702: Error deleting service during drain, review log files
- UPG-1703: Error creating service during drain, review log files
- UPG-1704: Error commenting service during drain, review log files
- UPG-1705: Error changing service owner during drain, review log files
- UPG-1706: Error starting service during drain, review log files
- UPG-1707: Error stopping service during drain, review log files
- UPG-1708: Error changing the source database state
- UPG-1709: Error serializing the database state
- UPG-1710: Error during PDB operation in drain
- UPG-1711: An error occurred while deferring Standby in Drain phase

- UPG-1800: Progress report is empty, review log file
- UPG-1900: Unable to create preupgrade directory
- UPG-1901: Error during user-defined preupgrade action
- UPG-2000: Creation of GRP failed
- UPG-2001: Unable to drop GRP
- UPG-2002: Restoration of the GRP failed

Non-CDB to CDB Upgrade Errors (UPG-3000 to

- UPG-3000: Error executing noncdbtopdb task
- UPG-3001: Could not describe the specified database
- UPG-3002: Could not execute database action
- UPG-3003: Plugin violations found
- UPG-3004: Could not create pluggable database
- UPG-3005: Error running noncdb_to_pdb.sql script
- UPG-3006: Error starting the database
- UPG-3007: User requested an abort of the noncdbtopdb task
- UPG-3008: Unable to remove the oratab entry of the source DB
- UPG-3009: After running noncdb_to_pdb.sql, the pdb was closed or was opened in restricted mode
- UPG-3010: Error running approot_to_pdb.sql script
- UPG-3100: Unable to list RAC service
- UPG-3101: Unable to shutdown RAC services
- UPG-3102: Unable to shutdown RAC database
- UPG-3103: Failed to disable RAC database
- UPG-3104: Unable to remove spfile parameter
- UPG-3105: Unable to start RAC database
- UPG-3106: Unable to upgrade RAC database on CRS
- UPG-3107: Unable to update spfile information on CRS
- UPG-3108: Failed to enable RAC database on CRS
- UPG-3109: Unable to stop current instance
- UPG-3110: Unable to downgrade RAC database source binaries on CRS
- UPG-3111: Failed to enable RAC database source binaries on CRS
- UPG-3112: Unable to start RAC database after downgrade source binaries on CRS
- UPG-3113: After executing "srvctl stop database", AutoUpgrade sees that the RAC database is still up, which suggests that the database was started up manually and not via CRS. This condition can cause AutoUpgrade to stop the current job.
- CON-4000: Database {0} shutdown or open with incorrect binaries for {1}. Ensure it is open with {2}
- CON-4001Database {0} currently has a status of {1}. For {2} mode, open it with one of the following: {3}

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Z