Oracle® Database

Using AutoUpgrade to Upgrade and Convert Non-CDBs to a PDB with the Same Operating System

21c
F48020-04
February 2024

ORACLE

Oracle Database Using AutoUpgrade to Upgrade and Convert Non-CDBs to a PDB with the Same Operating System, 21c

F48020-04

Primary Author: Douglas Williams

Contributing Authors: Daniel Overby Hansen, Nirmal Kumar, Sunil Surabhi

Contributors: Mark Bauer, Mike Dietrich, Rajesh Bhatiya, Hector Vieyra Farfan, Prakash Jashnani, Cindy Lim, Byron Motta, Padmaja Potineni, Roy Swonger, Carol Tagliaferri, Zhihai Zhang

Contents

Preface

This guide provides a compilation of topics from the Oracle Database user assistance documentation that are collected to help you complete a specific use case scenario.

Use Case Scenario for this Document

Use this scenario document to assist you to upgrade and convert to a PDB an earlier release non-CDB to the new release Oracle Database with the AutoUpgrade utility.

Prerequisites for this Scenario

- You have installed a new release of Oracle Database, and you have created a new container database or have an existing container database that can be used.

Oracle recommends that you back up your database.

Outline for this Scenario

1. **Checking Compatibility Before Upgrading Oracle Database**. Check that your earlier release is compatible with this upgrade scenario.
2. **Preparing to Upgrade Oracle Database**. Review steps and complete preparation tasks for this upgrade scenario.
3. **Upgrading Oracle Database**. Upgrade and convert your database from a non-CDB to a PDB on a multitenant Oracle Database using the AutoUpgrade utility.
4. **Post-upgrade tasks for Oracle Database**. Complete this basic list of post-upgrade tasks.

These steps correspond to the chapters in this document.

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.

1

Checking Compatibility Before Upgrading Oracle Database

Check the Oracle Database server upgrade compatibility matrix before upgrading the Oracle Database.

- Oracle Database Releases That Support Direct Upgrade
  Review the supported options for direct upgrades to Oracle Database 21c.
- Checking the Compatibility Level of Oracle Database
- Values for the COMPATIBLE Initialization Parameter in Oracle Database

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.

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';
```

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

Table 1-2 (Cont.) The COMPATIBLE Initialization Parameter

Oracle Database Release	Default Value	Minimum Value
Oracle Database 12c Release 2 (12.2)	12.2.0	11.2.0

2
Preparing to Upgrade Oracle Database

Before you upgrade Oracle Database, review new features, and carry out procedures to prepare your database for upgrade.

> **Note:**
>
> Oracle strongly recommends that you test the upgrade process and prepare a backup strategy.

- Installing Oracle Software in a New Oracle Home
  Choose a new location for the target Oracle home, and then install the new Oracle Database release software for single-instance.
- Prepare a Backup Strategy Before Upgrading Oracle Database Using AutoUpgrade
  You must design and carry out an appropriate backup strategy to ensure a successful upgrade.
- Database Preparation Tasks to Complete Before Starting Oracle Database Upgrades
  Ensure that you have completed these database preparation tasks before starting an Oracle Database upgrade.
- Preparing for 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.
- Preparations for Running AutoUpgrade Processing Modes
  You must complete preparations before you can run an AutoUpgrade processing mode.
- 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.
- 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.
- 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.

- 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.
- 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.
- 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.
- Examples of Non-CDB to PDB Configuration Files for AutoUpgrade
  Use these examples to understand how you can modify your own Oracle Database upgrade configuration file for AutoUpgrade.

Prepare a Backup Strategy Before Upgrading Oracle Database Using AutoUpgrade

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. For unplug-plug upgrades, remove it entirely, or 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

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 2-1 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)

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. You can also review the checklist on My Oracle Support, but ensure that you download the latest version of the AutoUpgrade tool, and use the checklist AutoUpgrade generates for your upgrade.

Related Topics

- My Oracle Support AutoUpgrade Tool (Doc ID 2485457.1)
- Upgrade your Database – NOW! Mike Dietrich's Oracle Database Upgrade Blog
- Database Preupgrade tool (via autoupgrade.jar) check list (Doc ID 2380601.1)

Installing Oracle Software in a New Oracle Home

Choose a new location for the target Oracle home, and then install the new Oracle Database release software for single-instance.

- Choose a New Location for Oracle Home when Upgrading or Patching
- Installing the New Oracle Database Software for Single Instance

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

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.

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
- Understanding Password Case Sensitivity and Upgrades
- Checking for Accounts Using Case-Insensitive Password Version
- Running Upgrades with Read-Only Tablespaces
  To take user schema-based tablespaces offline during upgrade, use AutoUpgrade with the `catctl_options` parameter `-T` option.

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

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 2-2 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 2-3 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 2-4 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*

Running Upgrades with Read-Only Tablespaces

To take user schema-based tablespaces offline during upgrade, use AutoUpgrade with the `catctl_options` parameter `-T` option.

For all Oracle Database releases that AutoUpgrade can upgrade, Autoupgrade can read file headers created in earlier releases. 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 `dbupgrade` log directory, and run commands in the log files manually to bring up tablespaces. You can view tablespace commands in the following log files:

- Non-CDB Upgrade format: `catupgrd`*`YYYYMMDDHHMMSC`*`0.log`, where:

  `YYYY` is the year, `MM` is the month, `DD` is the day, `HH` is the hour, `MM` is the minute in the hour, and `SC` is the seconds.

- PDB databases format: `catupgrd`*`YYYYMMDDHHSCpdbname`*`0.log`, where:

  `YYYY` is the year, `MM` is the month, `DD` is the day, `HH` is the hour, `MM` is the minute in the hour, `SC` is the seconds, and *`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

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

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 23c 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)

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/21.0.0/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`.
- 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.
- 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.
- 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
```

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
```

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
```

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

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
```

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 an unplug-relocate upgrade.
- 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.
- 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.
- 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_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.
- 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.
- 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_hcheck
  (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. This parameter is mandatory when you want to upgrade and convert a non-CDB Oracle Database.
- 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`.

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` 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 an unplug-relocate upgrade.

Usage Notes

During an unplug-relocate operation, 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
```

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=*
```

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_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.

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=[STANDBYS=NONE|STANDBYS=ALL|
STANDBYS=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')
```

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=*
```

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

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.

Example

```
sales1.raise_compatible=yes
```

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 you set `restoration=no`, then both the database backup and restoration must be performed manually. Use this option for databases that operate in `NOARCHIVELOG` mode, and for Standard Edition and 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_hcheck

(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_HCHECK` PL/SQL package on the source database as part of preupgrade

checks. If set, the AutoUpgrade `run_hcheck` 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_HCHECK`.

Example

```
sales1.run_hcheck=full
sales2.run_hcheck=critical
```

Related Topics

- DBMS_HCHECK

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 and user 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.

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.

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

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 targetr 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 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 concatinate 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`
`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`
`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	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_pdbs_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=proactive_fixups=true,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
```

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.
- 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

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.auto_log_dir:different/path
```

The result of using this syntax is that log files and temporary files are placed in the following path for databases identified by the prefix *`myprefix`*:

```
/path/to/my/global/log/dir/different/path
```

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.
- `-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
```

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.

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

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
```

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.

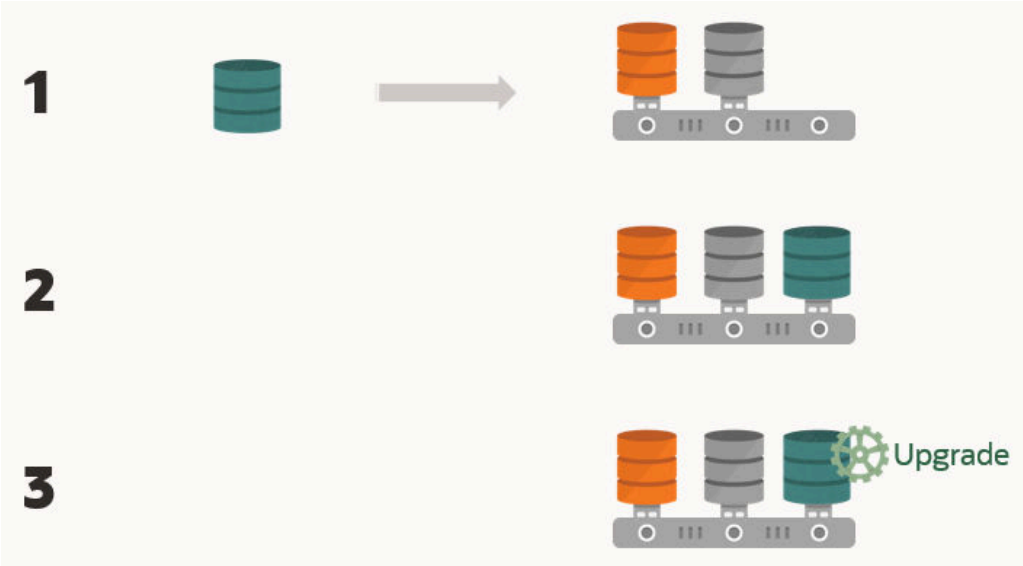
The non-CDB to PDB feature of the AutoUpgrade utility provides you flexible options to control how you upgrade your earlier release non-CDB Oracle Database when you upgrade and convert to the multitenant architecture. Starting with Oracle Database 21c, when you have an existing target release CDB, you 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 2-1 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 2-5 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

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 2-6 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.

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

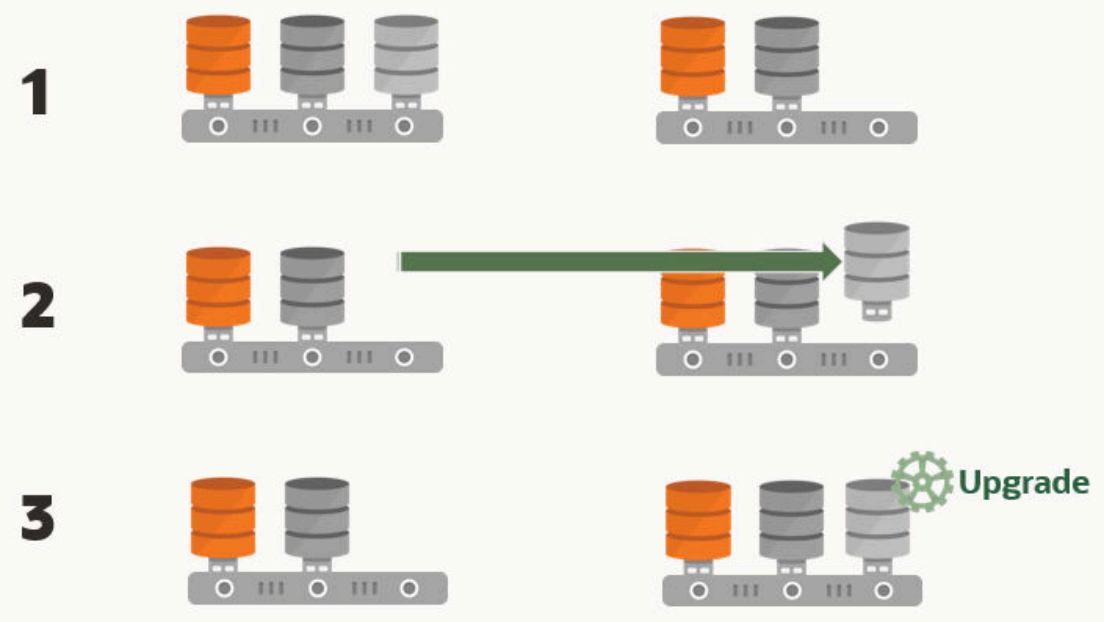
In addition, for unplug-plug operations, AutoUpgrade now supports moving the default state of a PDB from the source PDB to the target PDB. If you set `alter pluggable database save state` on a source PDB, then that state is transferred to the target PDB, so that the PDB is automatically opened when CDB$ROOT is opened.

> **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 2-2 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 2-7 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')
```

Examples of Non-CDB to PDB Configuration Files for AutoUpgrade

Use these examples to understand how you can modify your own Oracle Database upgrade configuration file for AutoUpgrade.

These examples are for an upgrade from an Oracle Database 12c Release 2 (12.2) non-CDB named DB12 to an Oracle Database 19c PDB named PDB3 in the target Oracle Database 19c CDB named CDB2. To understand details of how the global and local parameters are used, refer to the parameter references.

> **Caution:**
>
> Because this upgrade is a conversion from a non-CDB to a PDB, AutoUpgrade cannot create a guaranteed restore point that enables you to restore the non-CDB. To ensure your ability to recover from an issue, either back up your earlier release database, or convert the CDB to a PDB in your earlier release Oracle Database, and then upgrade and convert the earlier release PDB to the later release.

Example 2-8 AutoUpgrade Configuration File for Upgrade and Convert with Separate Backup Solution for Source Database

In this example, the configuration file directs AutoUpgrade to upgrade and convert the non-CDB Oracle Database 12c named DB12 to a PDB named PDB3 on the Oracle Database 21c CDB named CDB2.

```
global.autoupg_log_dir=/home/oracle/logs
upg1.source_home=/u01/app/oracle/product/12
upg1.target_home=/u01/app/oracle/product/21
upg1.sid=DB12
upg1.target_cdb=CDB2
upg1.target_pdb_name=PDB3
```

Example 2-9 AutoUpgrade Using `target_pdb_copy_option`

In this example, the parameter `upg1.target_pdb_copy_option` is used to have AutoUpgrade make a copy of the Oracle Database 12c (12.2.0.1) release to a PDB named PDB3, plugged

into the Oracle Database 19c CDB1. AutoUpgrade then moves PDB3 from `/u02/oradata/CDB1/pdb3` to `/u02/oradata/CDB2/pdb3`.

```
global.autoupg_log_dir=/home/oracle/logs

upg1.source_home=/u01/app/oracle/product/12.2.0.1
upg1.target_home=/u01/app/oracle/product/19
upg1.sid=CDB1
upg1.pdb=PDB3
upg1.target_cdb=CDB2
upg1.target_pdb_copy_option=file_name_convert=('CDB1', 'CDB2')
```

3

Using AutoUpgrade to Upgrade and convert Non-CDBs to PDBs

The AutoUpgrade Utility simplifies the task of upgrading and converting your earlier release Oracle Database to a later Oracle Database release using the multitenant architecture.

- 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.

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
   ```

4
Post-Upgrade Tasks for Oracle Database

After you have finished upgrading Oracle Database, complete the required post-upgrade tasks and consider these recommendations for the new release.

- Check the Upgrade With Post-Upgrade Status Tool
  Review the upgrade spool log file and use the Post-Upgrade Status Tool, `utlusts.sql`.
- 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.

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.

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.

> **Note:**
>
> This list of required tasks is based on the assumption that you have used AutoUpgrade to perform the upgrade. AutoUpgrade completes automatically many tasks that otherwise you are required to perform manually.

- 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.

- 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.
- Configuring the FTP and HTTP Ports and HTTP Authentication for Oracle XML DB
  Oracle Database Configuration Assistant (DBCA) does not configure ports for Oracle XML DB on Oracle Database 12c and later releases. Upgrades use digest authentication.
- Install Oracle Text Supplied Knowledge Bases After Upgrading Oracle Database
  After an Oracle Database upgrade, all user extensions to the Oracle Text supplied knowledge bases must be regenerated.
- 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.

Oracle provides the recompilation scripts `utlrp.sql`, `utlprp.sql`, and `utlprpom.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

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*

Configuring the FTP and HTTP Ports and HTTP Authentication for Oracle XML DB

Oracle Database Configuration Assistant (DBCA) does not configure ports for Oracle XML DB on Oracle Database 12c and later releases. Upgrades use digest authentication.

Oracle recommends that when you configure ports, you also configure the authentication for HTTP for accessing Oracle XML DB Repository to take advantage of improved security features.

Starting with Oracle Database 12c, Oracle enhanced database security by supporting digest authentication. Digest authentication is an industry-standard protocol that is commonly used with the HTTP protocol. It is supported by most HTTP clients. Digest authentication ensures that passwords are always transmitted in a secure manner, even when an encrypted (HTTPS) connection is not in use. Support for digest authentication enables organizations to deploy applications that use Oracle XML DB HTTP, without having to worry about passwords being compromised. Digest authentication support in Oracle XML DB also ensures that the Oracle XML DB HTTP server remains compatible with Microsoft Web Folders WebDAV clients.

After installing or upgrading for the new release, you must manually configure the FTP and HTTP ports for Oracle XML DB as follows:

1. Use `DBMS_XDB_CONFIG.setHTTPPort(`*`HTTP_port_number`*`)` to set the HTTP port for Oracle XML DB:

   ```
   SQL> exec DBMS_XDB_CONFIG.setHTTPPort(port_number);
   ```

2. Use `DBMS_XDB_CONFIG.setFTPPort(`*`FTP_port_number`*`)` to set the FTP port for Oracle XML DB:

   ```
   SQL> exec DBMS_XDB_CONFIG.setFTPPort(FTP_port_number);
   ```

   > **Note:**
   >
   > You can query the port numbers to use for FTP and HTTP in the procedure by using `DBMS_XDB_CONFIG.getFTPPort` and `DBMS_XDB_CONFIG.getHTTPPort` respectively.

3. To see all the used port numbers, query `DBMS_XDB_CONFIG.usedport`.

Install Oracle Text Supplied Knowledge Bases After Upgrading Oracle Database

After an Oracle Database upgrade, all user extensions to the Oracle Text supplied knowledge bases must be regenerated.

Regenerating the user extensions affect all databases installed in the given Oracle home.

After an upgrade, the Oracle Text-supplied knowledge bases that are part of the companion products for the new Oracle Database are not immediately available. Any Oracle Text

features dependent on the supplied knowledge bases that were available before the upgrade do not function after the upgrade. To re-enable such features, you must install the Oracle Text supplied knowledge bases from the installation media for the new Oracle Database release.

See Also:

- *Oracle Text Application Developer's Guide* for information about Oracle Text-supplied knowledge bases
- *Oracle Database Installation Guide* for companion products

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 4-1 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.
- 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 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 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 4-2 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. 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

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 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 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, and the desupport of Traditional Auditing in 23c, Oracle recommends that you transition to Unified Auditing. Oracle recommends that you turn off traditional audit in your database and use only unified auditing.

- 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

- How the Unified Auditing Migration Affects Individual Audit Features

Understanding Auditing for Oracle Database

Decide which audit policies you want to use in the upgraded database.

For newly created databases, unified auditing is enabled by default. The predefined audit policies `ORA_SECURECONFIG` and `ORA_LOGIN_LOGOUT` policies are enabled out-of-box.

> **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 23c.
>
> 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 My Oracle Support Doc ID 2369172.1, "LOB Columns of Database Audit Trails should use Securefile Storage" 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 non-unified auditing.

- *Oracle Database Security Guide*: This guide is the main source of information for configuring auditing. You must use the Oracle Database Release 11g version of this manual. To access this guide:

  1. Visit the database page on `docs.oracle.com` site on Oracle Technology Network:

     `https://docs.oracle.com/en/database/index.html`

  2. Select **Oracle Database**.
  3. In the Downloads page, select the **Documentation** tab.
  4. On the release list field, select **Earlier Releases**, and select Oracle Database 11g Release 2 (11.2).
  5. From the Oracle Database 11g Release 2 (11.2) Documentation page, select the **All Books** link to display publications in the documentation set.
  6. Search for *Security Guide*.
  7. Select either the **HTML** or the **PDF** link for this guide.

Identify Oracle Text Indexes for Rebuilds

You can run a script that helps you to identify Oracle Text index indexes with token tables that can benefit by being rebuilt after upgrading to the new Oracle Database release..

When you upgrade from Oracle Database 12c release 1 (12.2.0.1) to Oracle Database 18c and later releases, the Oracle Text token tables (`$I`, `$P`, and so on) are expanded from 64 bytes to 255 bytes. However, if you have indexes with existing token tables using the smaller size range, then the Oracle Text indexes cannot take advantage of this widened token column range. You must rebuild the indexes to use the 255 byte size range. Oracle provides a script that can assist you to identify indexes that can benefit by being rebuilt.

Obtain the script from My Oracle Support:

https://support.oracle.com/rs?type=doc&id=2287094.1

Dropping and Recreating DBMS_SCHEDULER Jobs

If DBMS_SCHEDULER jobs do not function after upgrading from an earlier release, drop and recreate the jobs.

If you find that DBMS_SCHEDULER jobs are not functioning after an upgrade. drop and recreate those jobs. This issue can occur even if the upgrade process does not report issues, and system objects are valid.

Transfer Unified Audit Records After the Upgrade

Review these topics to understand how you can obtain better performance after you upgrade and migrate to unified auditing

- About Transferring Unified Audit Records After an Upgrade
  Transferring the unified audit records from Oracle Database 12c release 12.1 to the new relational table under the `AUDSYS` schema for the new Oracle Database release improves the read performance of the unified audit trail.
- Transferring Unified Audit Records After an Upgrade
  You can transfer unified audit records to the new relational table in `AUDSYS` by using the `DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS` PL/SQL procedure.

About Transferring Unified Audit Records After an Upgrade

Transferring the unified audit records from Oracle Database 12c release 12.1 to the new relational table under the `AUDSYS` schema for the new Oracle Database release improves the read performance of the unified audit trail.

Starting with Oracle Database 12c Release 2, unified audit records are written directly to a new internal relational table that is located in the `AUDSYS` schema. In Oracle Database 12c release 12.1, the unified audit records were written to the common logging infrastructure (CLI) SGA queues. If you migrated to unified auditing in that release, then to obtain better read performance, you can transfer the unified audit records that are from that release to the new Oracle Database release internal table. It is not mandatory that you perform this transfer, but Oracle recommends that you do so to obtain better unified audit trail read performance. This is a one-time operation. All new unified audit records that are generated after the upgrade are written to the new table. The table is a read-only table. Any attempt to modify the metadata or data of this table is mandatorily audited.

After you upgrade to the new Oracle Database release, if you have any unified audit records present in the UNIFIED_AUDIT_TRAIL from the earlier release, then consider transferring them to the new internal relational table by using the transfer procedure for better read performance of the unified audit trail.

As with the SYS schema, you cannot query the AUDSYS schema if you have the SELECT ANY TABLE system privilege. In addition, this table is not listed as a schema object in the ALL_TABLES data dictionary view unless you have either the SELECT ANY DICTIONARY system privilege or an explicit SELECT privilege on this internal table. Until the database is open read write, the audit records are written to operating system spillover files (.bin format). However, you can transfer the audit records in these operating system files to the internal relational table after the database opens in the read write mode by using the DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES procedure.

Transferring Unified Audit Records After an Upgrade

You can transfer unified audit records to the new relational table in AUDSYS by using the DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS PL/SQL procedure.

1. Log in to the database instance as a user who has been granted the AUDIT_ADMIN role.

   For example, in a non-multitenant environment:

   ```
   sqlplus sec_admin
   Enter password: password
   ```

   For a multitenant environment, connect to the root:

   ```
   sqlplus c##sec_admin@root
   Enter password: password
   ```

   You can perform this procedure execution in the root as well as in a PDB, because the UNIFIED_AUDIT_TRAIL view is container specific. In addition, the transfer procedure is container specific. That is, performing the transfer from the root does not affect the unified audit records that are present in the unified audit trail for the PDB.

2. For a multitenant environment, query the DBA_PDB_HISTORY view to find the correct GUID that is associated with the CLI table that is specific to the container from which audit records must be transferred.

   For example:

   ```
   SQL> SELECT PDB_NAME, PDB_GUID FROM DBA_PDB_HISTORY;

   PDB_NAME  PDB_GUID
   --------  --------------------------------
   HR_PDB    33D96CA7862D53DFE0534DC0E40A7C9B
   ...
   ```

3. In a multitenant environment, connect to the container for which you want to transfer the audit records.

   You cannot perform the transfer operation on a container that is different from the one in which you are currently connected.

4. Run the DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS procedure.

   For example:

```
SQL> EXEC DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS;

PL/SQL procedure successfully completed.
```

Or, to specify the PDB GUID:

```
SQL> EXEC DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS
('33D96CA7862D53DFE0534DC0E40A7C9B');

PL/SQL procedure successfully completed.
```

5. If the database is in open read write mode, then run 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 run 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.

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.