Oracle® Database

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

19c
F10903-05
June 2021

Oracle Database Using AutoUpgrade to Upgrade and Convert Non-CDBs to a PDB with the Same Operating System, 19c

F10903-05

Primary Authors: Nirmal Kumar, Sunil Surabhi, Douglas Williams

Contributing Authors: Padmaja Potineni, Rajesh Bhatiya, Prakash Jashnani, Mark Bauer

Contributors: Roy Swonger, Byron Motta, Hector Vieyra Farfan, Daniel Overby Hansen, Carol Tagliaferri, Mike Dietrich, Marcus Doeringer, Umesh Aswathnarayana Rao, Rae Burns, Subrahmanyam Kodavaluru, Cindy Lim, Amar Mbaye, Akash Pathak, Thomas Zhang, 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
- Documentation Accessibility

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 Oracle Database release on your server with a multitenant container database (CDB) deployment.

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 19c 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.

- Checking the Compatibility Level of Oracle Database
- Values for the COMPATIBLE Initialization Parameter in Oracle Database

Checking the Compatibility Level of Oracle Database

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

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

Values for the COMPATIBLE Initialization Parameter in Oracle Database

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

Default and Minimum COMPATIBLE Parameter Values

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

Table 1-1 The COMPATIBLE Initialization Parameter

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

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 Oracle Home and then install the new Oracle Database Software for single-instance.
- Prepare a Backup Strategy Before Upgrading Oracle Database
  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.
- Enabling Oracle Database Vault After Upgrading Oracle Database
  Depending on your target database release, you can be required to disable Oracle Database Vault to complete an Oracle Database upgrade.
- 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 on a physical server or virtual machine, 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.
- Local Parameters for the AutoUpgrade Configuration File
  To configure information for specific Oracle Databases for the AutoUpgrade utility upgrade, you provide information in the AutoUpgrade local parameters.
- Global Parameters for the AutoUpgrade User Configuration File
  To specify a default behavior for a parameter for all Oracle Database upgrades addressed in the configuration file, you can use the optional AutoUpgrade global parameters.
- Locally Modifiable Global Parameters for AutoUpgrade Configuration File
  Locally modifiable global parameters are parameters that you set both globally, and as you require, set locally, so that you can better control AutoUpgrade job processing.

- 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.

Pre-Upgrade Information Check with AutoUpgrade

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

Before starting your upgrade, ensure that you have a new release Oracle Database installed and configured that you can use as the target for your upgrade. When your target Oracle Database home is prepared, run AutoUpgrade with the -preupgrade clause on your system, using the instructions that you can find in this guide.

Oracle requires that you run 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 Oracle’s upgrade blog for tips and suggestions that can assist you with your upgrade preparations.

Related Topics

- Upgrade your Database – NOW! Mike Dietrich's Oracle Database Upgrade Blog

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

After the upgrade, for each PDB, you must configure database listeners and local naming parameters (`tnsnames.ora` files). You must also configure Oracle Wallet management.

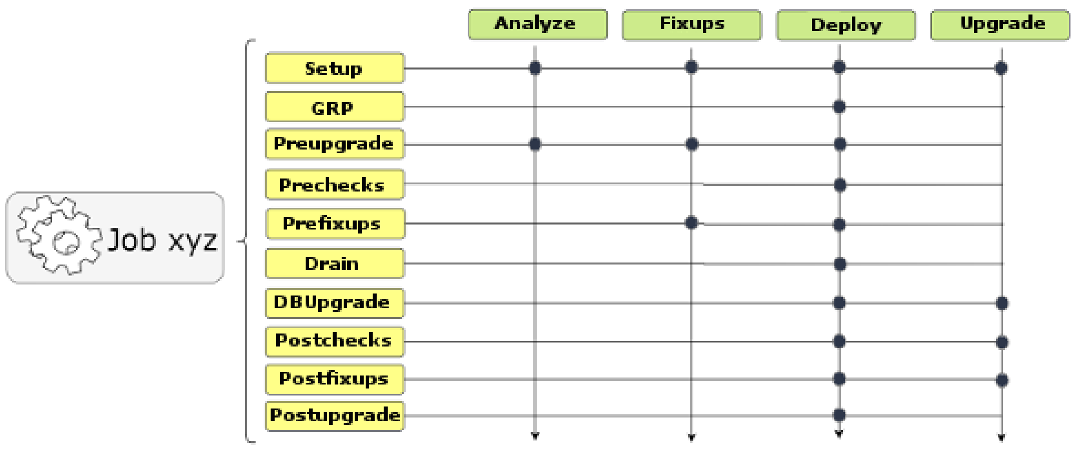
Caution:

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. AutoUpgrade does not support unplug-plug upgrades of PDBs that use Transparent Data Encryption (TDE), or that have an encrypted tablespace.

The following illustration shows the two unplug-plug options you can use:

1. Unplug PDBs from one source Oracle Database 12.2 CDB (`CDB1`, with `pdba` and `pdbb`) and plug them into a new release target CDB (`CDB3`).
2. Unplug PDBs from multiple source CDBs (Oracle Database 12.2 on `CDB1`, `pdba` and `pdbb`), and Oracle Database 18c, `CDB2`, `pdbc` and `pdbd`), and plug them into a new release target CDB (`CDB3`).

Figure 2-1 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 Application Express 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.

Caution:

Do not use AutoUpgrade to perform an unplug-plug upgrade to a CDB that is part of an Oracle Data Guard configuration. To upgrade a PDB using an unplug-plug to a CDB with an Oracle Data Guard configuration, you must perform the upgrade manually using the procedure described in the following My Oracle Support note:

Making Use Deferred PDB Recovery and the STANDBYS=NONE Feature with Oracle Multitenant (Doc ID 1916648.1)

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 2-1 AutoUpgrade Configuration File for Unplug-Plug Upgrades

To use the unplug-plug PDB upgrade option, you must set the system identifier parameters for the source CDB and PDB, parameter `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
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_version=19.1.0
upg1.target_cdb=cdb21x
upg1.target_pdb_name.pdb_2=depsales
upg1.target_pdb_copy_option.pdb_2=file_name_convert=('pdb_2','depsales')
```

Installing Oracle Software in a New Oracle Home

Choose a new location for Oracle Home and then install the new Oracle Database Software for single-instance.

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

Choose a New Location for Oracle Home when Upgrading

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

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

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

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

Installing the New Oracle Database Software for Single Instance

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

Note:

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

To install the new Oracle Database software for this release:

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

   When installation of Oracle Database software has completed successfully, click **Exit** to close Oracle Universal Installer.
3. 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.

Prepare a Backup Strategy Before Upgrading Oracle Database

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

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

To develop a backup strategy, consider the following questions:

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

Your backup strategy should answer all of these questions, and include procedures for successfully backing up and recovering your database. For information about implementing backup strategies using RMAN, review *Oracle Database Backup and Recovery User's Guide*.

Related Topics

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

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
- Recommendations for Oracle Net Services When Upgrading Oracle Database
- Understanding Password Case Sensitivity and Upgrades

- Checking for Accounts Using Case-Insensitive Password Version
- Running Upgrades with Read-Only Tablespaces

Release Updates and Requirements for Upgrading Oracle Database

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

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

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

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

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

Related Topics

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

Recommendations for Oracle Net Services When Upgrading Oracle Database

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

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

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

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

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

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

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

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

Related Topics

- *Oracle Database Net Services Reference*

Understanding Password Case Sensitivity and Upgrades

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

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

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

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

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

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

prevents the use of the case-sensitive password versions (the `11G` and `12C` password versions) for authentication.

Options for Accounts Using Case-Insensitive Versions

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

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

Related Topics

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

Checking for Accounts Using Case-Insensitive Password Version

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

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

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

Example 2-2 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

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

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

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

Viewing Tablespace Commands in Upgrade Log Files

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

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

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

```
SQL> ALTER TABLESPACE ARGROTBLSPA6 READ ONLY;

Tablespace altered.

SQL> ALTER TABLESPACE ARGROTBLSPB6 READ ONLY;

Tablespace altered.
```

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

```
SQL> ALTER TABLESPACE ARGROTBLSPA6 READ WRITE;

Tablespace altered.

SQL> ALTER TABLESPACE ARGROTBLSPB6 READ WRITE;

Tablespace altered.
```

See Also:

Oracle Database Administrator’s Guide for information about transporting tablespaces between databases

Enabling Oracle Database Vault After Upgrading Oracle Database

Depending on your target database release, you can be required to disable Oracle Database Vault to complete an Oracle Database upgrade.

- Upgrading Oracle Database Without Disabling Oracle Database Vault
- Common Upgrade Scenarios with Oracle Database Vault

Upgrading Oracle Database Without Disabling Oracle Database Vault

If your target Oracle Database release is 12.2 or later, then you can upgrade without disabling Oracle Database Vault.

If you have Oracle Database Vault enabled in your source Oracle Database release, then you can upgrade Oracle Database to Oracle Database 18c and later releases without first disabling Oracle Database Vault. After the upgrade, if your source Oracle Database release is Oracle Database 12c release 1 (12.1) or later, then Oracle Database Vault is enabled with the same enforcement settings that you had in place before the upgrade. For example, if your source database is Oracle Database release 12.1, and Oracle Database Vault was disabled in that release, then it remains disabled after you upgrade. If your source Oracle Database release 12.1 database had Oracle Database Vault enabled before the upgrade, then Oracle Database Vault is enabled after the upgrade.

If you manually disable Oracle Database Vault before the upgrade, then you must enable Oracle Database Vault manually after the upgrade.

If you did not have Oracle Database Vault enabled before the upgrade, then you can enable it manually after the upgrade.

Enable Oracle Database Vault in the upgraded database by using the procedure `dvsys.dbms_macadm.enable_dv()`. Run this procedure with a user account that is granted `DV_OWNER`. After you run the procedure, restart the database instance so that the procedure takes effect.

Related Topics

- *Oracle Database Vault Administrator’s Guide*

Common Upgrade Scenarios with Oracle Database Vault

The requirements to enable Oracle Database Vault after upgrades change, depending on your source Oracle Database release.

- Upgrades from Oracle Database 11*g* release 2 (11.2) or earlier: After the upgrade, Oracle Database Vault is disabled by default.
- Upgrades from Oracle Database 12*c* release 1 (12.1) or later: After the upgrade, Oracle Database Vault has the same enforcement status that you had in place before the upgrade.

Table 2-1 Common Oracle Database Vault Upgrade Scenarios and Upgrade Preparation Tasks

Source Database Release	Target Database Release	Do you need to disable Database Vault Before Upgrade	What is Database Vault Status After Upgrade
11.2 or earlier	12.1	Yes	Disabled. You need to enable Database Vault manually after the upgrade.

Table 2-1 (Cont.) Common Oracle Database Vault Upgrade Scenarios and Upgrade Preparation Tasks

Source Database Release	Target Database Release	Do you need to disable Database Vault Before Upgrade	What is Database Vault Status After Upgrade
11.2.or earlier	12.2, 18.1 and later	No	Disabled. You need to enable Database Vault manually after the upgrade.
12.1, 12.2, 18.1, and later	12.2, 18.1 and later	No	Database Vault has the same enforcement status that you had in place before the upgrade.

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.

In this example, user `oracle` navigates to the location of an earlier release Oracle home, which in this case is Oracle Database 12c Release 2 (12.2):

```
cd /u01/app/oracle/product/12.2/
```

Next, the Oracle user starts AutoUpgrade from the Oracle Database 19c Oracle home, and creates a configuration file in its user home directory, `/home/oracle`:

```
java -jar /u01/app/oracle/product/19/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
```

Local Parameters for the AutoUpgrade Configuration File

To configure information for specific Oracle Databases for the AutoUpgrade utility upgrade, you provide information in the AutoUpgrade local parameters.

Usage Notes

Local parameters take precedence over any global parameters set in the AutoUpgrade configuration file. Except where indicated with (Optional), all local parameters are required. All local parameters take a prefix (in examples, identified by a value you define to identify a particular database or upgrade. The prefix identifies the specific upgrade job to which the parameter applies in the configuration file.

Example: The set of parameters for the first upgrade in the configuration file uses the prefix *`sales`*, and the set of parameters for the next upgrade in the configuration file uses the prefix *`employees`*:

```
sales.source_home=/u01/app/oracle/12.2/dbhome1
.
.
```

```
.
employees.sid=salescdb
employees.source_home-/03/app/oracle/21/dbhome1
```

Table 2-2 Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`add_after_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to add after the upgrade. Example: `sales3.add_after_upgrade_pfile=/path/to/my/pfile_add.ora`
`add_during_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to add during the upgrade. Example: `sales3.add_during_upgrade_pfile=/path/to/my/newpfile.ora`

`after_action`	(Optional) Specifies a custom action that you want to have performed after completing the upgrade job for the database identified by the prefix address.
	The script that you use must be in the form of *`name.ext`* (for example, `myscript.sh`, so that AutoUpgrade can identify the type of script that you want to run. Permitted extension options:
	• Unix shell (`.sh`)
	• Microsoft Windows batch (`.bat`, `.cmd`)
	• Microsoft Windows PowerShell (`.ps1`)
	• Oracle SQL file (`.sql`), with a local operation only designated by the prefix.
	By default, if the script fails, then AutoUpgrade continues to run. Use the `Y` flag to specify that AutoUpgrade stops if the operating system detects that your script fails. If the script finishes with a status different than `0`, then it is considered a failed completion.
	In contrast to the global `after_action` parameter, the local `after_action` parameter can specify a SQL script, which then runs on the database using the target Oracle Database binaries on a non-CDB Oracle home, or on `CDB$ROOT`. If you want to run additional container-specific actions, then they must be set within the code. For more complex scenarios, you can run container-specific actions in a shell.
	Examples:
	Run the specified script before AutoUpgrade starts processing, with the Y flag set to stop AutoUpgrade if the script fails:
	`sales2.after_action=/user/path/script.sh Y`
	Run the specified script before AutoUpgrade starts processing, with AutoUpgrade set to continue to run if the script fails:
	`sales3.after_action=/user/path/script.sh`

Table 2-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`before_action`	(Optional) Specifies a custom action that you want to have performed before starting the upgrade job for the specific database job addressed by the prefix. If you want to have a script run before all upgrade jobs, then specify that script by using the local parameter (`global.before_action`)
	
	The script that you use must be in the form of *`name.ext`* (for example, `myscript.sh`), so that AutoUpgrade can identify the type of script that you want to run. Permitted extension options:
	• Unix shell (`.sh`)
	• Microsoft Windows batch (`.bat`, `.cmd`)
	• Microsoft Windows PowerShell (`.ps1`)
	• Oracle SQL file (`.sql`), with a local operation only designated by the prefix.
	
	By default, if the script fails, then AutoUpgrade continues to run. Use the `Y` flag to specify that AutoUpgrade stops if the operating system detects that your script fails. If the script finishes with a status different than `0`, then it is considered a failed completion.
	
	In contrast to the global `before_action` parameter, the local `before_action` parameter can specify a SQL script, which can run on the database in the source database Oracle home, using the earlier release Oracle Database binaries. The script runs on a non-CDB Oracle home, or on `CDB$ROOT`. If you want to run additional container-specific actions, then they must be set within the code. For more complex scenarios, you can run container-specific actions in a shell.
	
	Examples:
	
	Run the specified script before AutoUpgrade starts processing, with the `Y` flag set to stop AutoUpgrade if the script fails:
	
	`sales.before_action=/user/path/script.sh Y`
	
	Run the specified script before AutoUpgrade starts processing, with AutoUpgrade set to continue to run if the script fails:
	
	`sales4.before_action=/user/path/script.sh`

Table 2-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`catctl_options`	(Optional) Specifies one or more of a set of `catctl.pl` options that you can select for AutoUpgrade to submit for `catctl.pl` to override default behavior. For a complete description of the options, refer to "Parallel Upgrade Utility (`catctl.pl`) Parameters," which is linked to at the end of this table. Available `catctl.pl` options: • `-n` Number of processes to use for parallel operations. • `-N` Number of SQL processors to use when upgrading PDBs. • `-t` Run SQL in classic upgrade overwriting default replay upgrade method • `-T` Takes offline user schema-based table spaces. • `-z` Turns on production debugging information for `catcon.pm`. Example: `sales4.catctl_options=-t`

`checklist`	(Optional) Specifies the path to a checklist that you can use to override the default list of fixups that AutoUpgrade performs, such as fixups that you do not want implemented automatically, due to policy or security concerns.
	To use this parameter during other AutoUpgrade modes, you must run AutoUpgrade in `analyze` mode. After AutoUpgrade finishes the analysis, you can then find the checklist file identified by the database name under the precheck directory (*`dbname`*`_checklist.cfg`). Update the file manually to exclude the fixups that you want AutoUpgrade to bypass, and save the file with a new name. When you run AutoUpgrade again, you can specify the parameter pointing to the checklist file that you created, and modify fixups that are performed for individual databases. If you do not specify a checklist file path, then the set of fixups that run during the upgrade is the latest version of the checklist file that is created during the deploy mode that you specified.
	Example:
	`sales.checklist=/u01/app/oracle/upgrade-jobs/`
	`salesdb_checklist.cfg`
	In the preceding example, `salesdb_checklist.cfg` is the checklist configuration file for the database `salesdb`.

`del_after_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to remove after the upgrade. Example: `sales3.del_after_upgrade_pfile=/path/to/my/pfile_del.ora`

Table 2-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`del_during_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to have removed during upgrade. Example: `sales3.del_during_upgrade_pfile=/path/to/my/oldpfile.ora`

`env`	(Optional) Specifies custom operating system environment variables set on your operating system, excluding `ORACLE_SID`, `ORACLE_HOME`, `ORACLE_BASE`, and `TNS_ADMIN`.
	Use case:
	Use this parameter to provide environment setting that are indicated in the database `sqlnet.ora` file, such as secure socket layer cipher suites that are used for Oracle Wallet.
	Syntax:
	`*prefix*.env=VARIABLE1=value1/, VARIABLE2=value2/`
	For example, assume that for the PDB `sales2`, the value for `WALLET_LOCATION` is set using custom environment variables:
	`WALLET_LOCATION=`
	`  (SOURCE=`
	`    (METHOD=file)`
	`    (METHOD_DATA=(DIRECTORY=/databases/`
	`wallets/$CUSTOM_ENV1/$CUSTOM_ENV2))`
	In that case, for AutoUpgrade to know what those custom environment variables are, you must provide them using the `env` parameter, where *`dir1`* is the path indicated by the environment variable `CUSTOM_ENV1`, and *`dir2`* is the path specified by `CUSTOM_ENV2`:
	`sales2.env=CUSTOM_ENV1=*dir1*,CUSTOM_ENV2=*dir2*`

Table 2-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`log_dir`	(Optional with AutoUpgrade 19.8) Sets the location of log files that are generated for database upgrades that are in the set of databases included in the upgrade job identified by the prefix for the parameter.
	When set, AutoUpgrade creates a hierarchical directory based on a local log file path that you specify. For example, where the job identifier prefix is `sales`, and where `log_dir` is identified as `upgrade-jobs`, and *`stage1`*, *`stage2`*, and *`stagen`* represent stages of the upgrades:
	`/u01/app/oracle/upgrade-jobs`
	`    /temp/`
	`    /sales/`
	`    /sales/stage1`
	`    /sales/stage2`
	`    /sales/stagen`
	You cannot use wild cards for paths, such as tilde (~). You must use a complete path.
	Example:
	`salesdb.log_dir=/u01/app/oracle/upgrade-jobs`
	By default, if the global configuration file parameter `global.autoupg_log_dir` is specified, and you do not specify `log_dir`, then the default is the path specified in `global.autoupg_log_dir`.
	When neither `global.autoupg_log_dir` nor `log_dir` is specified, then by default the log files are placed in the location indicated by the `orabase` utility for the databases that you include in your configuration file. In that case, the default logs directory is in the path *`ORACLE_BASE`*`/cfgtoollogs/autoupgrade`.
	If the `orabase` utility fails for all databases included in the configuration file, then the log file location is then based on the `temp` directory for the user running AutoUpgrade.

`pdbs`	(Optional) Sets a list of PDBs on which you want the upgrade to run. This parameter only applies to upgrades of multitenant architecture (CDB) databases. If you are plugging in and upgrading a non-CDB database, then this parameter is ignored. The PDB list is comma-deliminated. The list can contain either PDB names, or a star character (`*`), which indicates that you want to upgrade all PDBs that are open on the CDB at the time that you run AutoUpgrade. If a PDB is in a mounted state, then AutoUpgrade ignores that PDB when running in ANALYZE mode. If the parameter is not specified, then all PDBs in the CDB are upgraded. However, if the PDB is in mount state, and the deploy mode is fixups, deploy or upgrade, then the PDB is opened in read-write mode, or upgrade mode, or both, so that the stages can run. Example: `sales1.pdbs=pdb1, pdb2, pdb`*`n`* `   `*`upgr1`*`.pdbs=*`

Table 2-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`raise_compatible`	(Optional) Increases the compatible parameter to the default value of the target release after the upgrade is completed successfully.
	Options:
	`[yes | no]`
	The default value is `no`.
	CAUTION:
	• After the `COMPATIBLE` parameter is increased, database downgrade is not possible.
	• Oracle recommends that you only raise the `COMPATIBLE` parameter to the current release level after you have thoroughly tested the upgraded database.
	• Regardless of what value you use for the `autoupgrade` command-line parameter `restore`, if you set the value of the configuration file parameter `raise_compatible` to `yes`, then before starting the upgrade, you must delete manually any guaranteed restore point you have created. After the upgrade is completed successfully, AutoUpgrade deletes the guaranteed restore point it creates before starting the upgrade. When AutoUpgrade starts the POSTUPGRADE stage, there is no way to restore the database.
	Example:
	`sales1.raise_compatible=yes`

`remove_underscore_parameters`	(Optional) Removes underscore (hidden) parameters from `PFILE` files during upgrade, and after upgrade, for all Oracle Databases in the configuration file. Underscore parameters should only be used by advice of Oracle Support.
	Options:
	`[yes | no]`
	The default value is `no`.
	Example:
	`sales1.remove_underscore_parameters=yes`

`restoration`	(Optional) Generates a Guaranteed Restore Point (GRP) for database restoration. If you set `restoration=no`, then both the database backup and restoration must be performed manually. Use this option for databases that operate in `NOARCHIVELOG` mode, and for Standard Edition and SE2 databases, which do not support the Oracle Flashback technology feature Flashback Database.
	Options:
	`[yes | no]`
	The default value is `no`.
	Example:
	`sales1.restoration=no`

Table 2-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`run_utlrp`	(Optional) Enables or disables running `utlrp` as part of the upgrade. The `utlrp` utility recompiles all Data Dictionary objects that become invalid during a database upgrade. Oracle recommends that you run this utility after every Oracle Database upgrade. Options: `yes`, `no`. The default is enabled (`yes`). Example: `prefix.run_utlrp=yes`
`sid`	(Required) Identifies the Oracle system identifier (SID) of the database that you want to upgrade. Example: `sales1.sid=salesdb`
`skip_tde_key_import`	(Optional) The default is `NO`. You can use this option for nonCDB-to-PDB and unplug/plug operations. When set to `YES`, the import of the source database KeyStore import into the target database is skipped, without raising an error. AutoUpgrade will leave the PDB open in upgrade mode, so that you can import the keys manually yourself.
`source_home`	(Required for `analyze`, `fixups`, and `deploy` modes. Optional for upgrade mode.) Current Oracle home (`ORACLE_HOME`) of the database that you want to upgrade. For the mode `upgrade`, the source home and target home values can be the same path. Example: `sales2.source_home=`*`/path/to/my/source/oracle/home`*
`source_tns_admin_dir`	(Optional) Specifies the path to the `TNS_ADMIN` directory in the source database home. This parameter has no effect on Microsoft Windows, because on Windows, the `TNS_ADMIN` environmental variable is set within the registry. Example: `sales1.source_tns_admin_dir=/u01/app/oracle/12.2/ dbhome01/network/admin`

Table 2-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`start_time`	(Optional) Sets a future start time for the upgrade job to run. Use this parameter to schedule upgrade jobs to balance the load on your server, and to prevent multiple jobs from starting immediately.
	Values must either take the form `now` (start immediately), or take the English Date Format form *DD/MM/YYYY* or *MM/DD/YYYY*, where *MM* is month, *DD* is day, and *YYYY* is year. If you do not set a value, then the default is `now`.
	Example:
	`sales1.start_time=now`
	`sales2.start_time=07/11/2020 01:30:15`
	Permitted values:
	`now`
	`30/12/2019 15:30:00`
	`01/11/2020 01:30:15`
	`2/5/2020 3:30:50`
	If more than one job is started with the `start_time` value set to `now`, then AutoUpgrade schedules start times based on resources available in the system, which can result in start time for jobs that are separated by a few minutes.
	Values are invalid that use the wrong deliminator for the date or time element, or use the wrong date or hour format.
	Example:
	`30-12-2019 15:30:00`
	`01/11/2020 3:30:15pm`
	`2020/06/01 01:30:15`

`target_base`	(Optional) Specifies the target `ORACLE_BASE` path for the target Oracle home.
	Example:
	`target_base=/u01/app/oracle`
	`sales4.target_base=/u04/app/oracle4`

`target_cdb`	(Optional) Specifies the SID of the target CDB into which a non-CDB Oracle Database is plugged in. This parameter is mandatory when you want to upgrade and convert a non-CDB Oracle Database. Example: `emp.target_cdb=salescdb`

Table 2-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`target_pdb_copy_option=file_name_convert=('f1', 'r1', 'f2', 'r2', 'f3', 'r3'...)`	(Optional) This option is only used when creating a pluggable database within the target CDB. It specifies the `file_name_convert` option that will be used by the create pluggable database statement that is executed by AutoUpgrade when converting a non-CDB database to a PDB or an existing PDB from a different source CDB into a PDB in the specified target CDB. If you do not specify this parameter, then the default value of the parameter is `NOCOPY`, and existing data files are reused.
	On the target CDB, if you have the parameters `DB_CREATE_FILE_DEST` or `PDB_FILE_NAME_CONVERT` set, and you want these parameters on the target CDB to take effect, then set the value of *`prefix`*`.target_pdb_copy_option=file_name_convert=NONE`
	If you want one or more data file names changed during conversion, then enter values for the parameter to indicate the filename you want to change, and the filename to which you want the existing files copied, using the syntax *`prefix`*`.target_pdb_copy_option=('`*`f1`*`', '`*`r1`*`', '`*`f2`*`', '`*`r2`*`', . . .)`, where *`f1`* is the first filename pattern on your source, *`r1`* is the first replacement filename pattern on your target CDB, *`f2`* is the second filename pattern on your source, *`r2`* is the second replacement file pattern on your target CDB, and so on.
	Example:
	In this example, AutoUpgrade will copy existing datafiles during conversion of a database specified with the prefix string `upg1` to replace the file path string and filename `/old/path/pdb_2` with the file path string and filename `/new/path/depsales`:
	`upg1.target_pdb_copy_option=file_name_convert=('/old/path/pdb_2', '/new/path/depsales')`
	To convert OMF files with `target_pdb_copy_option=file_name_convert`, the target Oracle home must be Oracle Database 19c Release Update 6 or later (19.6.0), or Oracle Database 18c Release Update 10 or later (18.10.0).
	In this example, the parameter is configured so that data files that are stored on Oracle ASM, but not stored as Oracle-managed files, are copied from `+DATA/dbname/sales` to `+DATA/dbname/depsales`:
	`upg1.target_pdb_copy_option=file_name_convert=('+DATA/dbname/sales', '+DATA/dbname/depsales')`

`target_pdb_name`	(Optional) Specifies the name that you want to assign to a non-CDB source Oracle Database after is plugged in to the target CDB. The default value is to use the database unique name of the non-CDB Oracle Database. If you want to specify a name that is different from the existing name of the non-CDB when you plug it in to the CDB, then you must set this parameter. Example: `emp.target_pdb_name=sales2`

Table 2-2 (Cont.) Local Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`target_tns_admin_dir`	(Optional) Specifies the path to the TNS_ADMIN directory in the target database home. Example: `sales1.target_tns_admin_dir=/u01/app/oracle/19/dbhome01/network/admin`
`timezone_upg`	(Optional) Enables or disables running the time zone upgrade as part of the AutoUpgrade process. To preserve data integrity, Oracle recommends that you upgrade the time zone settings at the time of your database upgrade. In particular, upgrade the timezone when you have data that depend on the time zone, such as `timestamp with time zone` table columns. Note that this setting can be disabled by overwriting the fixup on the checklist file. Options: `yes`, `no`. The default is enabled (`yes`). Example: `sales1.timezone_upg=yes`

`upgrade_node`	(Optional) Specifies the node on which the current user configuration is valid. The default value is `localhost`.
	The purpose of this parameter is to prevent AutoUpgrade from processing databases that are listed in the configuration file that you use with AutoUpgrade, where the value for the `upgrade_node` parameter does not correspond to the current host name. It does not enable running AutoUpgrade remotely. You can use the keyword `localhost` as a wild card to indicate that databases on the local host should be processed.
	Use case:
	The configuration file `config.cfg` contains 10 databases. Five of the databases have the value of `upgrade_node` set to `denver01`. The remaining five have the value of `upgrade_node` set to `denver02`. If AutoUpgrade is run on the server `denver01` using the configuration file `config.cfg`, then AutoUpgrade only processes the databases where `upgrade_node` is set to `denver01`. It ignores the databases where `upgrade_node` is set to `denver02`. The utility `hostname` identifies the value used to resolve the upgrade node.
	Example:
	`hostname`
	`denver02`
	`sales1.upgrade_node=denver01`

Global Parameters for the AutoUpgrade User Configuration File

To specify a default behavior for a parameter for all Oracle Database upgrades addressed in the configuration file, you can use the optional AutoUpgrade global parameters.

Usage Notes

All global parameters are optional. All global parameters take the prefix `global`.

The `add_after_upgrade_pfile` and `del_during_upgrade_pfile` global and local `PFILE` parameters operations are run in the following hierarchical order:

1. Global Actions
   a. Remove global
   b. Add global
2. Local Actions
   a. Remove local
   b. Add local

Table 2-3 Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description
`add_after_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to add after the `PFILE` is upgraded. This specification applies to all databases in the user configuration file. Example: `global.add_after_upgrade_pfile=/path/to/my/`*`add_after.ora`*
`add_during_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to have added during the `PFILE` upgrade. This specification applies to all databases in the user configuration file. `global.add_during_upgrade_pfile=/path/to/my/`*`add_during.ora`*

Table 2-3 (Cont.) Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`after_action`	(Optional) Specifies a path and a file name for a custom user script that you want to have run after all the upgrade jobs finish successfully. The script that you use must be in the form of *`name.ext`* (for example, `myscript.sh`, so that AutoUpgrade can identify the type of script that you want to run. Permitted extension options:
	• Unix shell (`.sh`)
	• Microsoft Windows batch (`.bat`, `.cmd`)
	• Microsoft Windows PowerShell (`.ps1`)
	By default, if the script fails, then AutoUpgrade continues to run. Use the `Y` flag to specify that AutoUpgrade stops if the operating system detects that your script fails. If the script finishes with a status different than `0`, then it is considered a failed completion.
	Examples:
	If the script fails, then stop AutoUpgrade:
	`global.after_action=/path/to/my/script.sh Y`
	If the script fails, then continue AutoUpgrade:
	`global.after_action=/path/to/my/script.sh`

`autoupg_log_dir`	(Optional) Sets the location of the log files, and temporary files that belong to global modules, which AutoUpgrade uses.
	Example:
	`global.autoupg_log_dir=/path/to/my/global/log/dir`
	Starting with AutoUpgrade 19.7, you can configure different log directory path in the `userconfig` file in the logs directory for a specific prefix
	`global.autoupg_log_dir=/path/to/my/global/log/dir`
	`myprefix.log_dir=global.auto_log_dir:different/path`
	The result of using this syntax is that log files and temporary files are placed in the following path for databases identified by the prefix *`myprefix`*:
	`/path/to/my/global/log/dir/different/path`
	If you do not set this parameter to a path, then by default the log files are placed in the location indicated by the `orabase` utility for the databases that you include in your configuration file. In that case, the default logs directory is in the path *`ORACLE_BASE`*`/cfgtoollogs/autoupgrade`.
	If the `orabase` utility fails for all databases included in the configuration file, then the log file location is then based on the `temp` directory for the user running AutoUpgrade.

Table 2-3 (Cont.) Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`before_action`	(Optional) Specifies a custom user script that you want to have run for all upgrades before starting the upgrade jobs. The script that you use must be in the form of *`name.ext`* (for example, `myscript.sh`), so that AutoUpgrade can identify the type of script that you want to run. If you want to have a script run before a specific upgrade job, then specify that script by using the local parameter (`local.before_action`)
	Permitted extension options:
	• Unix shell (`.sh`)
	• Microsoft Windows batch (`.bat`, `.cmd`)
	• Microsoft Windows PowerShell (`.ps1`)
	By default, if the script fails, then AutoUpgrade continues to run. Use the `Y` flag to specify that AutoUpgrade stops if the operating system detects that your script fails. If the script finishes with a status different than `0`, then it is considered a failed completion.
	Examples:
	If the script fails, then stop AutoUpgrade:
	`global.before_action=/path/to/my/script.sh Y`
	If the script fails, then continue AutoUpgrade:
	`global.before_action=/path/to/my/script.sh`

`catctl_options`	(Optional) Specifies one or more of a set of `catctl.pl` options that you can select for AutoUpgrade to submit for `catctl.pl` to override default behavior. For a complete description of the options, refer to "Parallel Upgrade Utility (`catctl.pl`) Parameters," which is linked to at the end of this table. Available `catctl.pl` options: • `-n` Number of processes to use for parallel operations. • `-N` Number of SQL processors to use when upgrading PDBs. • `-t` Run SQL in classic upgrade overwriting default replay upgrade method • `-T` Takes offline user schema-based table spaces. • `-z` Turns on production debugging information for `catcon.pm`. Example: `global.catctl_options=-t -n 24 -N 4`
`del_after_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to have removed after the `PFILE` upgrade. This specification applies to all databases in the user configuration file. Example: `global.del_after_upgrade_pfile=/path/to/my/del_after.ora`

Table 2-3 (Cont.) Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`del_during_upgrade_pfile`	(Optional) Specifies a path and file name of a `PFILE` whose parameters you want to have removed during the `PFILE` upgrade. This specification applies to all databases in the user configuration file.
	Example:
	`global.del_during_upgrade_pfile=/path/to/my/`
	`del_during.ora`

`drop_grp_after_upgrade`	(Optional) Deletes the Guaranteed Restore Point (GRP) after database upgrade. If you select this option, then GRP is deleted after upgrade completes successfully. If you set `raise_compatible` to `yes`, then you must also set the parameter `drop_grp_after_upgrade` to `yes`.
	Options:
	`[yes | no]`
	The default value is `no`.
	Example:
	`global.drop_grp_after_upgrade=yes`

`target_base`	(Optional) Specifies the target `ORACLE_BASE` path for the target Oracle home. Use of this parameter is only required in rare cases.
	Example:
	`global.target_base=/u01/app/oracle`
	`sales4.target_base=/u04/app/oracle4`

Table 2-3 (Cont.) Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`raise_compatible`	(Optional) Increases the compatible parameter to the default value of the target release after the upgrade is completed successfully.
	Options:
	`[yes | no]`
	The default value is `no`.
	CAUTION:
	• After the `COMPATIBLE` parameter is increased, database downgrade is not possible.
	• Oracle recommends that you only raise the `COMPATIBLE` parameter to the current release level after you have thoroughly tested the upgraded database.
	• Regardless of what value you use for the `autoupgrade` command-line parameter `restore`, if you set the value of the configuration file parameter `raise_compatible` to `yes`, then before starting the upgrade, you must delete manually any guaranteed restore point you have created. After the upgrade is completed successfully, AutoUpgrade deletes the guaranteed restore point it creates before starting the upgrade. When AutoUpgrade starts the POSTUPGRADE stage, there is no way to restore the database.
	• If you set `raise_compatible` to `yes`, then you must also set the parameter `drop_grp_after_upgrade` to `yes`.
	Example:
	`global.raise_compatible=yes`

`target_home`	(Optional for analyze and fixups modes. Required for upgrade and deploy modes.) Sets a global target home for all of the databases specified in the configuration file. Use this option to avoid specifying the same `target_home` multiple times. This parameter can be overwritten locally. Example: `global.target_home=/target/Oracle/home`

Table 2-3 (Cont.) Global Configuration Parameters for Oracle Database AutoUpgrade Utility

Parameter	Description

`target_version`	(Optional) Specifies the target release version on which you want AutoUpgrade to perform the upgrade. AutoUpgrade uses the release version information that you provide in this parameter to ensure that the correct checks and fixups are used for the target Oracle Database release to which you are upgrading. The format for this parameter are period-delimited values of valid Oracle versions.
	Valid values
	• 12.2
	• 18
	• 19
	This option is only required if the target home is not present on the system, or if the target home is a 12.2 release. Otherwise, AutoUpgrade can derive the target release value.
	Example:
	`global.target_version=18`
	`employees.target_version=12.2`

`upgradexml`	(Optional) Generates the `upgrade.xml` file. Options: `[yes | no]` The `upgrade.xml` is equivalent to the file in earlier releases that the preupgrade package generated when you specified the XML parameter. This file is created during the analyze mode (`mode -analyze`). It is generated in the prechecks directory defined for the AutoUpgrade log files. Example: `global.upgradexml=yes`

Related Topics

- Parallel Upgrade Utility (catctl.pl) Parameters

Locally Modifiable Global Parameters for AutoUpgrade Configuration File

Locally modifiable global parameters are parameters that you set both globally, and as you require, set locally, so that you can better control AutoUpgrade job processing.

Usage Notes

Locally modifiable global parameters are required parameters. You must define these parameters in your AutoUpgrade configuration file, either globally, or locally. With locally modifiable global parameters, you can use the prefix `global` to set values as global parameters for all jobs in your AutoUpgrade configuration file, but reset the same parameter with a local job prefix for a particular job in the same configuration file. You can also choose to set locally modifiable global parameters only as local parameters for each AutoUpgrade job.

Note:

These parameters are available in the latest version of AutoUpgrade that you can download from My Oracle Support.

When a locally modifiable global parameter is set both with a global prefix, and with a local job prefix, the locally modified parameter value overrides the global parameter values for the job identified by the prefix that you use with the parameter. The syntax you use is in the form `global.target_home=`*`Global target Oracle home`*, and *`database`*`.target_home=`*`local target Oracle home`*.

Example:

In the AutoUpgrade configuration file, the required parameter `target_home` is set globally to one Oracle home path. But in the configuration file, the same parameter is set locally to a different Oracle home path. As AutoUpgrade processes the jobs in the configuration file, it uses the locally defined path for `target_home` for the job defined by the prefix `upgrade3`, overriding the global parameter setting:

```
global.target_home=/u01/app/oracle/21.0.0/dbhome01
upgrade3.target_home=/u03/app/oracle3/12.2.0.1/dbhome3
```

Table 2-4 Locally Modifiable Global Parameters for AutoUpgrade Configuration Files

Parameter	Description
`defer_standby_log_shipping`	(Optional) Defers shipping logs from the primary database to the standby database before the upgrade, where you have a primary database with a physical standby database. When Autoupgrade defers log shipping, you will receive a notice that log shipping is deferred, and that after the upgrade completes successfully, you need to reenable shipping logs from the primary database to the secondary database. The default option is `No`. If you change the default to `Yes`, then log shipping is deferred.

`drop_grp_after_upgrade`	(Optional) Deletes the Guaranteed Restore Point (GRP) after database upgrade. If you select this option, then GRP is deleted after upgrade completes successfully.
	Options:
	`[yes | no]`
	The default value is `no`.
	Examples:
	`global.drop_grp_after_upgrade=yes`
	`sales.drop_grp_after_upgrade=yes`

Table 2-4 (Cont.) Locally Modifiable Global Parameters for AutoUpgrade Configuration Files

Parameter	Description
`enable_local_undo`	(Optional) For a CDB upgrade, specifies whether or not `LOCAL` undo should be enabled before the upgrade of `CDB$ROOT` by running the following statement: `ALTER DATABASE LOCAL UNDO ON;` The allowed values are [`YES \| NO`]. The default value is `NO`. When local undo is first enabled, the size of the undo tablespace in `PDB$SEED` is determined as a factor of the size of the undo tablespace in `CDB$ROOT`. The default is 30 percent of the undo tablespace size. Every other PDB in the CDB inherits this property from `PDB$SEED`. Ensure that there is enough space to allocate new `UNDO` tablespaces.
`manage_network_files`	Specifies whether network files are processed during the upgrade. Options: `[FULL|SKIP|IGNORE_READ_ONLY]` `FULL`: (default) Raise all exceptions encountered during the copy and merge of network files into the target Oracle home. `SKIP`: Do not process network files during postupgrade. `IGNORE_READ_ONLY`: Attempt to copy and merge network files, but do not raise an exception during the upgrade if the target file is read only The following network files are processed: `oranfstab`, `ldap.ora`, `tnsnames.ora`, `sqlnet.ora`, and `listener.ora`

`remove_underscore_parameters`	(Optional) Removes underscore (hidden) parameters from `PFILE` files during upgrade, and after upgrade, for all Oracle Databases in the configuration file. Underscore parameters should only be used by advice of Oracle Support.
	
	Options:
	
	`[yes | no]`
	
	The default value is `no`.
	
	Example:
	
	`global.remove_underscore_parameters=yes`

Table 2-4 (Cont.) Locally Modifiable Global Parameters for AutoUpgrade Configuration Files

Parameter	Description

`restoration`	(Optional, available with Enterprise Edition only) Generates a Guaranteed Restore Point (GRP) for database restoration. If you select this option, then both database backup and database restoration must be performed manually by the DBA.
	Options:
	`[yes | no]`
	The default value is `yes`.
	Example:
	`global.restoration=no`
	Standard Edition does not support Flashback Database, so this option is not available for Standard Edition. If your database is a Standard Edition Oracle Database, then you must ensure that you have a separate fallback mechanism is in place.

`target_version`	(Optional) Specifies the target release version on which you want AutoUpgrade to perform the upgrade. AutoUpgrade uses the release version information that you provide in this parameter to ensure that the correct checks and fixups are used for the target Oracle Database release to which you are upgrading. The format for this parameter are period-delimited values of valid Oracle versions.
	Valid values
	• 12.2
	• 18
	• 19
	• 21
	This option is only required if the target home is not present on the system, or if the target home is a 12.2 release. Otherwise, AutoUpgrade can derive the target release value.
	Example:
	`global.target_version=18`
	`employees.target_version=12.2`

`target_home`	Specifies the target Oracle home (`ORACLE_HOME`) path.
	Example:
	`global.target_home=/u01/app/oracle/21.0.0/dbhome01`
	`sales4.target_home=/u04/app/oracle4/21.0.0/`
	`dbhome04`
	If the mode is `ANALYZE` or `FIXUPS`, then the parameter `target_home` is optional.

Table 2-4 (Cont.) Locally Modifiable Global Parameters for AutoUpgrade Configuration Files

Parameter	Description

`target_base`	(Optional) Specifies the target `ORACLE_BASE` path for the target Oracle home.
	Example:
	`global.target_base=/u01/app/oracle`
	`sales4.target_base=/u04/app/oracle4`

Understanding Non-CDB to PDB Upgrades with AutoUpgrade

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

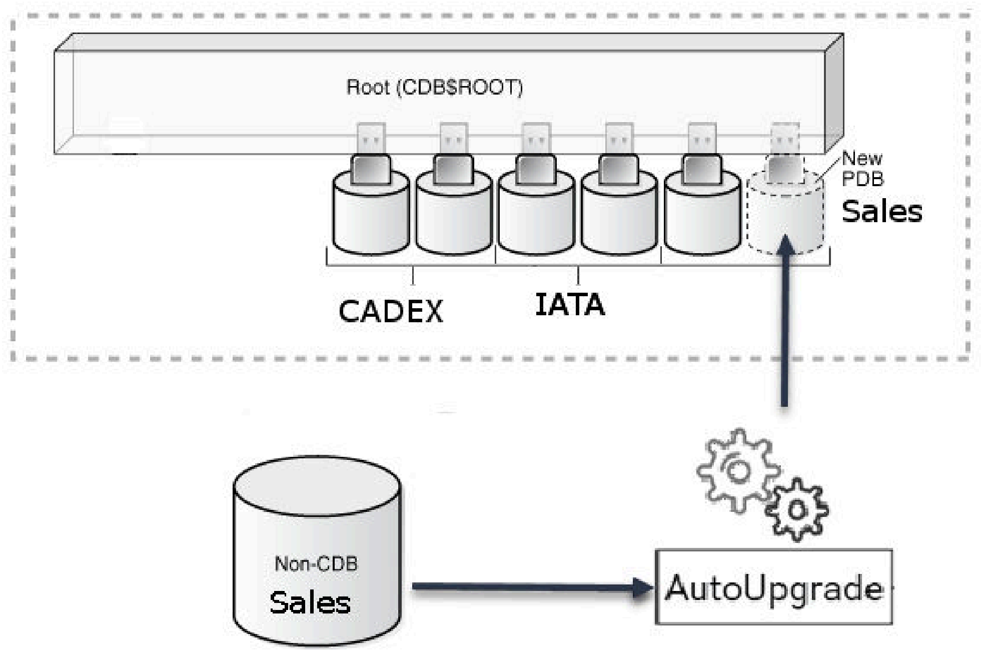
Oracle Database 19c is the terminal release in which non-CDB Oracle Database architecture is supported. Oracle strongly recommends that you move to using pluggable databases (PDBs). When you migrate your database from the non-CDB architecture to PDBs in Oracle Database 19c, you obtain up to three user-configurable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

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

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 2-2   Converting and Upgrading a Non-CDB Using AutoUpgrade



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/19.1.0/dbhome_1
upg1.target_base=/u01
upg1.target_version=19.1.0
upg1.target_cdb=cdb19x
```

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!*

Related Topics

- Unplug / Plug / Upgrade with AutoUpgrade in Mike Dietrich, Upgrade Your Database Now
- 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 no data is lost during the conversion, Oracle strongly recommends that allow time in your upgrade plan to implement your backup strategy before you use AutoUpgrade to perform a non-CDB upgrade and conversion.

Guidelines for Upgrade Planning

The non-CDB-to-PDB conversion and upgrade process is not recoverable. To ensure a proper upgrade and conversion, and to reduce unexpected downtime, Oracle strongly recommends that you address any error conditions found during the analyze phase.

If you do not set the `target_pdb_copy_option` in your AutoUpgrade configuration file, then the database conversion uses the same file location and file names that are used with existing database files. To prevent potential data loss, ensure that your data is backed up, and consider your file placement plans before starting AutoUpgrade.

GRP and Upgrades from Non-CDB to Multitenant Architecture

- During the upgrade, AutoUpgrade creates a guaranteed restore point (GRP) that is available only in the context of the upgrade stage of the AutoUpgrade Deploy workflow. To ensure against any potential data loss, you must implement your backup strategy before starting AutoUpgrade.
- Database conversion from non-CDB to the multitenant architecture is performed during the AutoUpgrade Drain stage. After this stage is complete, the GRP that AutoUpgrade creates is removed, and it is not possible to use the AutoUpgrade `restore` command to restore the database. In the event that you require a recovery to the earlier non-CDB Oracle Database release, you must be prepared to recover the database manually.

Example 2-6 Upgrading and Converting a Non-CDB to Oracle Database 19c Using Multitenant Architecture

During the Deploy conversion and upgrade workflow, AutoUpgrade version 19.9 and later creates a GRP, and runs the Prefixup stage. If any part of the Deploy workflow up to the Prefixup stage completion fails, then AutoUpgrade can restore the database back to the GRP created at the start of the deployment.

However, after the Prefixup stage is complete, the upgraded database is plugged in to the target release Oracle Database container database (CDB) to complete conversion. As soon as the non-CDB is plugged into the CDB, the GRP is no longer valid, and is dropped.

If anything goes wrong during the plug-in, then AutoUpgrade cannot recover and restore the database. You must restore the database manually.

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 19c. 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 2-7 AutoUpgrade Configuration File for Upgrade and Convert with Separate Backup Solutionfor 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 19c CDB named CDB2.

```
global.autoupg_log_dir=/home/oracle/logs
upg1.dbname=DB12
upg1.start_time=NOW
upg1.source_home=/u01/app/oracle/product/12
upg1.target_home=/u01/app/oracle/product/19
upg1.sid=DB12
upg1.log_dir=/home/oracle/logs
upg1.upgrade_node=localhost
upg1.target_version=19
upg1.restoration=no
upg1.target_cdb=CDB2
upg1.target_pdb_name=PDB3
```

Example 2-8 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')
upg1.log_dir=/home/oracle/logs
```

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 Oracle Database 19c 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. When relevant, customize the fixups that AutoUpgrade has created to address error conditions.

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

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

AutoUpgrade with Source and Target Database Homes on Different Servers

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

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

To start the analysis, enter the following command.

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

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

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

1. Run fixups on the source server:

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

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

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

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. You must complete these tasks both when you perform the upgrade with replay upgrade (the default) or with AutoUpgrade, except as noted.

- 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.
- Check PL/SQL Packages and Dependent Procedures
  It is possible that packages that you installed in the earlier release Oracle Database are not available in the new release, which can affect applications.
- Upgrading 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`.
- 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 disable Oracle Database Vault to complete an Oracle Database upgrade.
- Check for the SQLNET.ALLOWED_LOGON_VERSION Parameter Behavior
  Connections to Oracle Database from clients earlier than release 10g fail with the error `ORA-28040: No matching authentication protocol`.

Setting Environment Variables on Linux and Unix Systems After Manual Upgrades

Check that required operating system environment variables point to the directories of the new Oracle Database release.

Confirm that the following Oracle user environment variables point to the directories of the new Oracle home:

- `ORACLE_HOME`
- `PATH`

Related Topics

- Step 2: Ensure That the Required Environment Variables Are Set

Check PL/SQL Packages and Dependent Procedures

It is possible that packages that you installed in the earlier release Oracle Database are not available in the new release, which can affect applications.

After the upgrade, if you use AutoUpgrade, review the AutoUpgrade report on invalid objects. If you use a replay upgrade, then check to ensure that any packages that you may have used in your own scripts, or that you call from your scripts, are available in the new release. Testing procedures dependent on packages should be part of your upgrade plan.

Code in database applications can reference objects in the connected database. For example, Oracle Call Interface (OCI) and precompiler applications can submit anonymous PL/SQL blocks. Triggers in Oracle Forms applications can reference a schema object. Such applications are dependent on the schema objects they reference. Dependency management techniques vary, depending on the development environment. Oracle Database does not automatically track application dependencies.

Related Topics

- *Oracle Database Administrator's Guide*

Upgrading 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

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 disable Oracle Database Vault to complete an Oracle Database upgrade.

- Upgrading Oracle Database Without Disabling Oracle Database Vault
  If your target Oracle Database release is 12.2 or later, then you can upgrade without disabling Oracle Database Vault.
- Common Upgrade Scenarios with Oracle Database Vault
  The requirements to enable Oracle Database Vault after upgrades change, depending on your source Oracle Database release.

Upgrading Oracle Database Without Disabling Oracle Database Vault

If your target Oracle Database release is 12.2 or later, then you can upgrade without disabling Oracle Database Vault.

If you have Oracle Database Vault enabled in your source Oracle Database release, then you can upgrade Oracle Database to Oracle Database 18c and later releases without first disabling Oracle Database Vault. After the upgrade, if your source Oracle Database release is Oracle Database 12c release 1 (12.1) or later, then Oracle Database Vault is enabled with the same enforcement settings that you had in place before the upgrade. For example, if your source database is Oracle Database release 12.1, and Oracle Database Vault was disabled in that release, then it remains disabled after you upgrade. If your source Oracle Database release 12.1 database had Oracle Database Vault enabled before the upgrade, then Oracle Database Vault is enabled after the upgrade.

If you manually disable Oracle Database Vault before the upgrade, then you must enable Oracle Database Vault manually after the upgrade.

If you did not have Oracle Database Vault enabled before the upgrade, then you can enable it manually after the upgrade.

Enable Oracle Database Vault in the upgraded database by using the procedure `dvsys.dbms_macadm.enable_dv()`. Run this procedure with a user account that is granted `DV_OWNER`. After you run the procedure, restart the database instance so that the procedure takes effect.

Related Topics

- *Oracle Database Vault Administrator’s Guide*

Common Upgrade Scenarios with Oracle Database Vault

The requirements to enable Oracle Database Vault after upgrades change, depending on your source Oracle Database release.

- Upgrades from Oracle Database 11*g* release 2 (11.2) or earlier: After the upgrade, Oracle Database Vault is disabled by default.
- Upgrades from Oracle Database 12*c* release 1 (12.1) or later: After the upgrade, Oracle Database Vault has the same enforcement status that you had in place before the upgrade.

Table 4-1 Common Oracle Database Vault Upgrade Scenarios and Upgrade Preparation Tasks

Source Database Release	Target Database Release	Do you need to disable Database Vault Before Upgrade	What is Database Vault Status After Upgrade
11.2 or earlier	12.1	Yes	Disabled. You need to enable Database Vault manually after the upgrade.
11.2.or earlier	12.2, 18.1 and later	No	Disabled. You need to enable Database Vault manually after the upgrade.
12.1, 12.2, 18.1, and later	12.2, 18.1 and later	No	Database Vault has the same enforcement status that you had in place before the upgrade.

Check for the SQLNET.ALLOWED_LOGON_VERSION Parameter Behavior

Connections to Oracle Database from clients earlier than release 10g fail with the error `ORA-28040: No matching authentication protocol`.

Starting with Oracle Database 18c, the default value for the `SQLNET.ALLOWED_LOGON_VERSION` parameter changed from 11 in Oracle Database 12c (12.2) to 12 in Oracle Database 18c and later releases. The use of this parameter is deprecated.

`SQLNET.ALLOWED_LOGON_VERSION` is now replaced with the `SQLNET.ALLOWED_LOGON_VERSION_SERVER` and `SQLNET.ALLOWED_LOGON_VERSION_CLIENT` parameters. If you have not explicitly set the `SQLNET.ALLOWED_LOGON_VERSION_SERVER` parameter in the upgraded database, then connections from clients earlier than release 10g fail with the error `ORA-28040: No matching authentication protocol`. For better security, check the password verifiers of your database users, and then configure the database to use the correct password verifier by setting the `SQLNET.ALLOWED_LOGON_VERSION_SERVER` and `SQLNET.ALLOWED_LOGON_VERSION_CLIENT` parameters.

If you have password-protected roles (secure roles) in your existing database, and if you upgrade to Oracle Database 18c and later releases with the default `SQLNET.ALLOWED_LOGON_VERSION_SERVER` setting of 12, because those secure roles only have release 10g verifiers, then the password for each secure role must be reset by the administrator so that the secure roles can remain usable after the upgrade.

See Also:

- *Oracle Database Security Guide* for information about ensuring against password security threats
- *Oracle Database Security GuideOracle Database Security Guide* for information about setting the password versions of users

Recommended and Best Practices to Complete After Upgrading Oracle Database

Oracle recommends that you complete these good practices guidelines for updating Oracle Database. Except where noted, these practices are recommended for all 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.
- 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`.

- Migrate Your Upgraded Oracle Databases to Use Unified Auditing
  To use the full facilities of unified auditing, you must manually migrate to unified auditing.
- Identify Oracle Text Indexes for Rebuilds
  You can run a script that helps you to identify Oracle Text index indexes with token tables that can benefit by being rebuilt after upgrading to the new Oracle Database release..
- Dropping and Recreating DBMS_SCHEDULER Jobs
  If DBMS_SCHEDULER jobs do not function after upgrading from an earlier release, drop and recreate the jobs.
- Transfer Unified Audit Records After the Upgrade
  Review these topics to understand how you can obtain better performance after you upgrade and migrate to unified auditing
- About 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.
- About Testing the Upgraded Production Oracle Database
  Repeat tests on your production database that you carried out on your test database to ensure applications operate as expected.
- Upgrading the Time Zone File Version After Upgrading Oracle Database
  If the AutoUpgrade preupgrade report in `upgrade.xml` 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 the `DBMS_DST` PL/SQL package to upgrade the time zone file.

Back Up the Database

Oracle strongly recommends that you at least perform a level 1 backup, or if time allows, perform a level 0 backup.

Related Topics

- Backing Up the Database

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/19.0.0/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`.

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 a baseline that is useful for performance tuning, Oracle recommends that you gather baseline statistics at a point when the system is operating at an optimal level.

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. Case sensitivity increases the security of passwords by requiring that users enter both the correct password string, and the correct case for each character in that string. For example, the password `hPP5620qr` fails if it is entered as `hpp5620QR` or `hPp5620Qr`.

To secure your database, create passwords in a secure fashion. If you have default passwords in your database, then change these passwords. By default, case sensitivity is enforce when you change passwords. Every password should satisfy the Oracle recommended password requirements, including passwords for predefined user accounts.

For new databases created after the upgrade, there are no additional tasks or management requirements.

Existing Database Requirements and Guidelines for Password Changes

- If the default security settings for Oracle Database 12*c* release 1 (12.1) and later are in place, then passwords must be at least eight characters, and passwords such as `welcome` and `oracle` are not allowed.
- The `IGNORECASE` parameter is deprecated. Do not use this parameter.
- For existing databases, to take advantage of password case-sensitivity, you must reset the passwords of existing users during the database upgrade procedure. Reset the password for each existing database user with an `ALTER USER` statement.
- Query the `PASSWORD_VERSIONS` column of `DBA_USERS` to find the `USERNAME` of accounts that only have the 10G password version, and do not have either the `11G` or the `12C` password version. Reset the password for any account that has only the `10G` password version.

See Also:

- *Oracle Database Security Guide* for more information about password case sensitivity
- *Oracle Database Security Guide* for more information about password strength

Finding and Resetting User Passwords That Use the 10G Password Version

For better security, find and reset passwords for user accounts that use the `10G` password version so that they use later, more secure password versions.

Finding All Password Versions of Current Users

You can query the DBA_USERS data dictionary view to find a list of all the password versions configured for user accounts.

For example:

```
SELECT USERNAME,PASSWORD_VERSIONS FROM DBA_USERS;

USERNAME                       PASSWORD_VERSIONS
------------------------------ -----------------
JONES                          10G 11G 12C
ADAMS                          10G 11G
CLARK                          10G 11G
PRESTON                        11G
BLAKE                          10G
```

The PASSWORD_VERSIONS column shows the list of password versions that exist for the account. 10G refers to the earlier case-insensitive Oracle password version, 11G refers to the SHA-1-based password version, and 12C refers to the SHA-2-based SHA-512 password version.

- User jones: The password for this user was reset in Oracle Database 12*c* Release 12.1 when the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter setting was 8. This enabled all three password versions to be created.
- Users adams and clark: The passwords for these accounts were originally created in Oracle Database 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:

Oracle Database New Features Guide

Develop New Administrative Procedures as Needed

Plan a review of your scripts and procedures, and change as needed.

After familiarizing yourself with the features of the new Oracle Database release, review your database administration scripts and procedures to determine whether any changes are necessary.

Coordinate your changes to the database with the changes that are necessary for each application. For example, by enabling integrity constraints in the database, you may be able to remove some data checking from your applications.

Migrating Tables from the LONG Data Type to the LOB Data Type

You can use the `ALTER TABLE` statement to change the data type of a `LONG` column to `CLOB` and that of a `LONG RAW` column to `BLOB`.

The `LOB` data types (`BFILE`, `BLOB`, `CLOB`, and `NCLOB`) can provide many advantages over `LONG` data types.

In the following example, the `LONG` column named `long_col` in table `long_tab` is changed to data type `CLOB`:

```
SQL> ALTER TABLE Long_tab MODIFY ( long_col CLOB );
```

After using this method to change `LONG` columns to LOBs, all the existing constraints and triggers on the table are still usable. However, all the indexes, including Domain indexes and Functional indexes, on all columns of the table become unusable and must be rebuilt using an `ALTER INDEX...REBUILD` statement. Also, the Domain indexes on the `LONG` column must be dropped before changing the `LONG` column to a LOB.

See Also:

Oracle Database SecureFiles and Large Objects Developer's Guide for information about modifying applications to use LOB data

Migrate Your Upgraded Oracle Databases to Use Unified Auditing

To use the full facilities of unified auditing, you must manually migrate to unified auditing.

In unified auditing, all Oracle Database audit trails (`SYS.AUD$` for the database audit trail, `SYS.FGA_LOG$` for fine-grained auditing, `DVYS.AUDIT_TRAIL$` for Database Vault, and so on) are combined into one single audit trail, which you can view by querying the `UNIFIED_AUDIT_TRAIL` data dictionary view for single-instance installations and `GV$UNIFIED_AUDIT_TRAIL` for Oracle Real Application Clusters environments.

- Understanding Unified Auditing Migration Process for Oracle Database
  Decide which audit policies you want to use in the upgraded database.
- Migrating to Unified Auditing for Oracle Database
  Use this procedure for multitenant container (CDB) databases to migrate to unified auditing.
- About Managing Earlier Audit Records After You Migrate to Unified Auditing
  Review, archive, and purge earlier audit trails in preparation for using the unified audit trail.
- Removing the Unified Auditing Functionality
  Use this procedure to remove unified auditing, and to use mixed-mode audit.
- Obtaining Documentation References if You Choose Not to Use Unified Auditing
  You can access documentation listed here to obtain configuration information about how to use non-unified auditing.

See Also:

Oracle Database Security Guide for information about how the audit features have changed for this release

Understanding Unified Auditing Migration Process for Oracle Database

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

By default, unified auditing is not enabled for upgraded databases. If you have upgraded from an earlier release to Oracle Database 12*c*, then your database uses the same auditing functionality that was used in the earlier release. For newly created databases, the mixed-mode method of unified auditing is enabled by default. After you complete the migration to unified auditing, traditional auditing is disabled, and the new audit records write to the unified audit trail.

To enable and configure the audit policies and how they are used, choose one method as follows:

- Use the pure unified audit facility.

  Migrate to unified auditing to use the full unified auditing facility features. After you complete the procedure to migrate to unified auditing, you can create and enable new audit policies and also use the predefined audit policies. The audit records for these policies write to the unified audit trail. The earlier audit trails and their audit records remain, but no new audit records write to the earlier audit trails.

  Note:

  The audit configuration from the earlier release has no effect in the unified audit system. Only unified audit policies generate audit records inside the unified audit trail.

- Use a mixed-mode audit facility.

  The mixed-mode audit facility enables both traditional and unified auditing facilities to run simultaneously and applies to both new and upgraded databases. The mixed-mode unified auditing facility becomes available if you enable at least one of the unified auditing predefined audit policies. Audit records for these policies write to the unified audit trail. The audit configuration in the earlier release of Oracle Database is also available, and the audit records for this configuration write to the earlier audit trails. If you decide that you prefer using the pure unified audit facility, then you can migrate to it.

  Note:

  If the database is not writable, then audit records write to new format operating system files in the `$ORACLE_BASE/audit/$ORACLE_SID` directory.

  See Also:

  – *Oracle Database Security Guide* for information about the predefined audit policies

  – *Oracle Database Security Guide* for information about the `ora_SecureConfig` audit policy

Migrating to Unified Auditing for Oracle Database

Use this procedure for multitenant container (CDB) databases to migrate to unified auditing.

Perform the following procedure in the `root`. The procedure migrates both the `root` CDB, and any associated PDBs, to unified auditing.

Note:

You can disable unified auditing from the container database (CDB) root only, not for individual pluggable databases (PDBs).

However, when unified auditing is disabled, then 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 ioracle 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
   ```

About Managing Earlier Audit Records After You Migrate to Unified Auditing

Review, archive, and purge earlier audit trails in preparation for using the unified audit trail.

After you complete the procedure to migrate Oracle Database to use unified auditing, any audit records that your database had before remain in their earlier audit trails. You can archive these audit records and then purge their audit trails. With unified auditing in place, any new audit records write to the unified audit trail.

See Also:

- "Archiving the Audit Trail" in *Oracle Database Security Guide*
- "Purging Audit Trail Records" in *Oracle Database Security Guide*

Removing the Unified Auditing Functionality

Use this procedure to remove unified auditing, and to use mixed-mode audit.

After you have enabled your databases to use unified auditing, if you decide that you do not want unified auditing, then you can use this procedure to remove the unified auditing functionality. In this case, your database uses the mixed-mode audit facility.

1. Stop the database.

   ```
   sqlplus sys as sysoper
   Enter password: password

   SQL> SHUTDOWN IMMEDIATE
   SQL> EXIT
   ```

   For Windows systems, stop the Oracle service:

   ```
   net stop OracleService%ORACLE_SID%
   ```

   For Oracle RAC installations, shut down each database instance as follows:

   ```
   srvctl stop database -db db_name
   ```

2. Go to the `$ORACLE_HOME/rdbms/lib` directory.
3. Disable the unified auditing executable.
   - **Unix:** Run the following command:

     ```
     make -f ins_rdbms.mk uniaud_off ioracle ORACLE_HOME=$ORACLE_HOME
     ```

   - **Microsoft Windows:** Rename the `%ORACLE_HOME%/bin/orauniaud12.dll` file to `%ORACLE_HOME%/bin/orauniaud12.dll.dbl`.
4. Restart the database.

   ```
   sqlplus sys as sysoper
   Enter password: password
   ```

```
SQL> STARTUP
SQL> EXIT
```

For Microsoft Windows systems, start the Oracle service again.

```
net start OracleService%ORACLE_SID%
```

For Oracle RAC installations, start each database instance using the following syntax:

```
srvctl start database -db db_name
```

Obtaining Documentation References if You Choose Not to Use Unified Auditing

You can access documentation listed here to obtain configuration information about how to use non-unified auditing.

After upgrading to the new release Oracle Database, if you choose not to change to unified auditing, then Oracle documentation and Oracle Technology Network provide information about traditional non-unified auditing.

- *Oracle Database Security Guide*: This guide is the main source of information for configuring auditing. You must use the Oracle Database Release 11g version of this manual. To access this guide:
  1. Visit the database page on `docs.oracle.com` site on Oracle Technology Network:

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

Identify Oracle Text Indexes for Rebuilds

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

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

Obtain the script from My Oracle Support:

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

Dropping and Recreating DBMS_SCHEDULER Jobs

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

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

Transfer Unified Audit Records After the Upgrade

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

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

About Transferring Unified Audit Records After an Upgrade

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

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

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

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

Transferring Unified Audit Records After an Upgrade

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

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

   For example, in a non-multitenant environment:

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

   For a multitenant environment, connect to the root:

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

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

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

   For example:

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

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

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

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

4. Run the `DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS` procedure.

   For example:

   ```
   SQL> EXEC DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS;

   PL/SQL procedure successfully completed.
   ```

   Or, to specify the PDB GUID:

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

   PL/SQL procedure successfully completed.
   ```

5. If the database is in open read write mode, then execute the `DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES` procedure.

Until the database is in open read write mode, audit records are written to operating system (OS) files. The `DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES` procedure moves the unified audit records that are present in the files to database tables. You can find the unified audit records that are present in the OS spillover files by querying the `V$UNIFIED_AUDIT_TRAIL` dynamic view.

For example, if you want to execute this procedure for audit records in the `HR_PDB` container, then you must connect to that PDB first:

```
SQL> CONNECT sec_admin@HR_PDB
Enter password: password

SQL> EXEC DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES;

PL/SQL procedure successfully completed.
```

6. Query the `UNIFIED_AUDIT_TRAIL` data dictionary view to check if the records transferred correctly.

   Oracle highly recommends that you query `UNIFIED_AUDIT_TRAIL`. After a successful audit record transfer, you should query the `UNIFIED_AUDIT_TRAIL` because querying the `V$UNIFIED_AUDIT_TRAIL` dynamic view will show the audit records that are present only in the OS spillover files.

About Recovery Catalog Upgrade After Upgrading Oracle Database

If you use a version of the recovery catalog schema that is older than that required by the RMAN client, then you must upgrade it.

See Also:

- *Oracle Database Backup and Recovery User's Guide* for information on managing an RMAN recovery catalog
- *Oracle Database Backup and Recovery User's Guide* for complete information about upgrading the recovery catalog and the `UPGRADE CATALOG` command

About Testing the Upgraded Production Oracle Database

Repeat tests on your production database that you carried out on your test database to ensure applications operate as expected.

If you upgraded a test database to the new Oracle Database release, and then tested it, then you can now repeat those tests on the production database that you upgraded to the new Oracle Database release. Compare the results, noting anomalies. Repeat the test upgrade as many times as necessary.

To verify that your applications operate properly with a new Oracle Database release, test the newly upgraded production database with your existing applications. You also can test enhanced functions by adding available Oracle Database features, and then testing them. However, first ensure that the applications operate in the same manner as they did before the upgrade.

Upgrading the Time Zone File Version After Upgrading Oracle Database

If the AutoUpgrade preupgrade report in `upgrade.xml` 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 the `DBMS_DST` PL/SQL package to upgrade the time zone file.

Oracle Database supplies multiple versions of time zone files. There are two types of files associated with each time zone file: a large file, which contains all the time zones defined in the database, and a small file, which contains only the most commonly used time zones. The large versions are designated as `timezlrg_`*`version_number`*`.dat`. The small versions are designated as `timezone_`*`version_number`*`.dat`. The files are located in the `oracore/zoneinfo` subdirectory under the Oracle Database home directory.

Related Topics

- Upgrading Time Zone Data Using the DBMS_DST Package
- https://support.oracle.com/rs?type=doc&id=1585343.1