

# Perceptive Content Database Upgrade Package for Oracle Server

## Reference Guide

Version: Foundation 22.2

Written by: Documentation Team, R&D  
Date: June 2023

# Documentation Notice

Information in this document is subject to change without notice. The software described in this document is furnished only under a separate license agreement and may only be used or copied according to the terms of such agreement. It is against the law to copy the software except as specifically allowed in the license agreement. This document or accompanying materials may contain certain information which is confidential information of Hyland Software, Inc. and its affiliates, and which may be subject to the confidentiality provisions agreed to by you.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright law, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Hyland Software, Inc. or one of its affiliates.

Hyland, HXP, OnBase, Alfresco, Nuxeo, and product names are registered and/or unregistered trademarks of Hyland Software, Inc. and its affiliates in the United States and other countries. All other trademarks, service marks, trade names and products of other companies are the property of their respective owners.

© 2023 Hyland Software, Inc. and its affiliates.

The information in this document may contain technology as defined by the Export Administration Regulations (EAR) and could be subject to the Export Control Laws of the U.S. Government including for the EAR and trade and economic sanctions maintained by the Office of Foreign Assets Control as well as the export controls laws of your entity's local jurisdiction. Transfer of such technology by any means to a foreign person, whether in the United States or abroad, could require export licensing or other approval from the U.S. Government and the export authority of your entity's jurisdiction. You are responsible for ensuring that you have any required approvals prior to export.

## Table of Contents

|   |           |
|---|-----------|
| <b>Documentation Notice</b> .....                                     | <b>2</b>  |
| <b>Overview</b> .....   | <b>5</b>  |
| <b>Benefits</b> .....   | <b>5</b>  |
| <b>Summary of steps</b> .....   | <b>5</b>  |
| With uptime (EP2 7.5).....  | 6         |
| <i>Setup steps</i> .....  | 6         |
| <i>Uptime steps</i> .....   | 6         |
| <i>Downtime steps</i> .....   | 6         |
| Without uptime.....   | 6         |
| <i>Setup steps</i> .....  | 6         |
| <i>Downtime steps</i> .....   | 6         |
| Status procedures (EP2 7.5).....                                      | 7         |
| <b>Prerequisites</b> .....  | <b>7</b>  |
| Database storage for the IN_DOC table and indexes (EP2 7.5).....      | 7         |
| <i>Additional storage considerations</i> .....                        | 7         |
| Formatting procedure output.....                                      | 7         |
| Logging.....  | 7         |
| <b>Create the packages</b> .....                                      | <b>8</b>  |
| IN_DB_UTIL package.....   | 8         |
| IN_DB_UPGRADE package.....  | 8         |
| <b>Setup steps</b> .....  | <b>8</b>  |
| Create control tables.....  | 8         |
| Synchronization framework (EP2 7.5).....                              | 9         |
| <b>Synchronization steps (EP2 7.5)</b> .....                          | <b>9</b>  |
| <b>Uptime steps (EP2 7.5)</b> .....                                   | <b>10</b> |
| <b>Downtime steps</b> .....   | <b>10</b> |
| Pre-downtime steps.....   | 10        |
| Downtime steps.....   | 10        |
| Post downtime steps.....  | 11        |
| <b>View and update upgrade parameter default values</b> .....         | <b>11</b> |
| <b>View and update synchronization parameter default values</b> ..... | <b>12</b> |
| <b>Troubleshooting</b> .....  | <b>12</b> |
| <b>Cleaning up after an early exit</b> .....                          | <b>13</b> |

|   |           |
|---|-----------|
| <b>Upgrade and synchronization procedures</b> ..... | <b>14</b> |
| Upgrade setup procedures.....                       | 14        |
| <i>UPGRADE_SETUP</i> .....                          | 14        |
| Synchronization setup procedures (EP2 7.5).....     | 17        |
| <i>SYNC_SETUP</i> .....                             | 17        |
| <i>SYNC_SETUP_IN_DOC</i> .....                      | 19        |
| Synchronization procedures (EP2 7.5).....           | 20        |
| <i>SYNC_TABLE_IN_DOC</i> .....                      | 20        |
| <i>SYNC_GET_STATUS</i> .....                        | 21        |
| <i>SYNC_RESET</i> .....                             | 22        |
| <i>SYNC_REMOVE</i> .....                            | 24        |
| <i>SYNC_CHECK</i> .....                             | 25        |
| Upgrade procedures.....                             | 25        |
| <i>DOWNTIME_STEPS</i> .....                         | 25        |
| Upgrade procedures (EP2 7.5).....                   | 26        |
| <i>UPTIME_STEPS</i> .....                           | 26        |
| <i>UPTIME_GET_STATUS</i> .....                      | 28        |
| <i>DOWNTIME_STEPS</i> .....                         | 28        |
| <i>DOWNTIME_GET_STATUS</i> .....                    | 30        |
| <i>CREATE_DEST_TABLE_IN_DOC</i> .....               | 30        |
| <i>LOAD_DEST_TABLE_IN_DOC</i> .....                 | 31        |
| <i>CREATE_PK_IN_DOC</i> .....                       | 32        |
| <i>CREATE_INDEXES_IN_DOC</i> .....                  | 33        |
| <b>Upgrade objects</b> .....                        | <b>34</b> |
| IN_DB_UPGRADE_CONTROL_UPGRADE table.....            | 34        |
| <b>Synchronization objects (EP2 7.5)</b> .....      | <b>35</b> |
| IN_DB_UPGRADE_CONTROL_SYNC table.....               | 35        |
| SYNC_ERROR_DOC table.....                           | 36        |
| SYNC_HIST_DOC table.....                            | 36        |
| SYNC_STAGE_DOC table.....                           | 37        |
| SYNC_STATE_DOC table.....                           | 37        |
| SYNC_TRG_DOC trigger.....                           | 38        |
| SYNC_SEQ_DOC sequence.....                          | 38        |
| <b>Debugging</b> .....                              | <b>39</b> |
| <b>Synchronization benchmarks (EP2 7.5)</b> .....   | <b>39</b> |

## Overview

The following document guides you through the upgrade process using the new Perceptive Content upgrade packages for Oracle, IN\_DB\_UTIL and IN\_DB\_UPGRADE.

You can use the IN\_DB\_UPGRADE package as an alternative to the traditional database incremental scripts that were previously and still available for database schema upgrades.

This collection of procedures and functions (package) is designed to provide a means for you to upgrade your Perceptive Content database schema in a manner that facilitates the execution of the most time-consuming schema changes in uptime to reduce the duration of downtime events during database upgrades.

This package supports upgrading the inuser database schema to version 7.9.0.0 (22.2) or 7.7.0.0 (22.1 and EP4) or 7.5.0.0 (EP3 and EP2) from 7.7.0.0 or 7.5.0.0 or 7.4.0.3 or 7.2.3.0 or 7.2.0.0 or 7.1.5.2 or 7.1.4.0 or 7.1.3.3.

This upgrade package is ideal for customers with very large databases who might benefit from executing the long running operations in uptime, resulting in a much shorter downtime event during the upgrade.

For upgrades with a starting schema version less than 7.5.0.0, using this package to upgrade the database schema involves the duplication of the IN\_DOC table as well as several large indexes. This requires the allocation of additional storage for the duration of the upgrade, which can be reclaimed afterwards if necessary.

## Benefits

This upgrade provides the following benefits:

- This upgrade package provides for additional flexibility by facilitating an upgrade to 7.9.0.0 (22.2) or 7.7.0.0 (22.1 or EP4) or 7.5.0.0 (EP3 and EP2) from any previous version between 7.1.3.3 – 7.7.0.0.
- The 7.2.2.0 and 7.5.0.0 incremental database scripts include operations that rebuild the IN\_DOC table and its indexes and constraints. These operations are time consuming and result in an extended downtime for larger databases. This upgrade package provides a means for accomplishing these lengthy operations during uptime in the days or hours leading up to the downtime and results in a much shorter downtime event.
- This upgrade package provides you with more control with more control over incrementally completing the uptime steps over short periods of time by providing the MAX\_MINUTES parameter. This parameter helps limit the scope per execution of the uptime steps by defining an end time that prevents new tasks from starting.
- This upgrade package provides you with more control over database upgrade operations by surfacing parameters such as tablespace names and degree of parallelism (DOP) for both uptime and downtime operations.

## Summary of steps

The following section outlines the summary of steps described in this document.

### Notes

- Synchronization related steps and procedures currently only apply for upgrades with a starting schema version less than 7.5.0.0.

- There are no qualifying uptime steps for upgrades with a starting schema version of 7.5.0.0 and up.

## With uptime (EP2 7.5)

The following applies to upgrades starting on schema versions prior to 7.5.0.0.

### Setup steps

- Use the following procedure to setup the upgrade and create the IN\_DB\_UPGRADE\_CONTROL\_UPGRADE table, which contains the default parameter values used by the various upgrade procedures.

**Note** For upgrades starting on schema versions less than 7.5.0.0 the UPGRADE\_SETUP command calls the SYNC\_SETUP procedure to create the IN\_DB\_UPGRADE\_CONTROL\_SYNC table, which contains the default parameter values used by the various synchronization procedures.

```
EXECUTE IN_DB_UPGRADE.UPGRADE_SETUP
```

### Uptime steps

- Use the following procedure to execute qualifying uptime schema changes until all steps have completed. This only applies to upgrades with a starting schema version less than 7.5.0.0.

```
EXECUTE IN_DB_UPGRADE.UPTIME_STEPS
```

- Use the following procedure, as needed, to synchronize and propagate rows between the IN\_DOC and NEW\_DOC tables. This only applies to upgrades with a starting schema version less than 7.5.0.0.

```
EXECUTE IN_DB_UPGRADE.SYNC_TABLE_IN_DOC
```

### Downtime steps

- Use the following procedure to execute any remaining qualifying uptime steps and downtime schema changes until all steps have completed.

```
EXECUTE IN_DB_UPGRADE.DOWNTIME_STEPS
```

## Without uptime

### Setup steps

- Use the following procedure to setup the upgrade and create the IN\_DB\_UPGRADE\_CONTROL\_UPGRADE table, which contains the default parameter values used by the various upgrade procedures.

```
EXECUTE IN_DB_UPGRADE.UPGRADE_SETUP
```

### Downtime steps

- Use the following procedure to execute all qualifying uptime and downtime schema changes until all steps have completed.

```
EXECUTE IN_DB_UPGRADE.DOWNTIME_STEPS
```

## Status procedures (EP2 7.5)

- Use the following procedures to check the synchronization and uptime status. This only applies to upgrades with a starting schema version less than 7.5.0.0.

```
EXECUTE IN_DB_UPGRADE.SYNC_GET_STATUS('IN_DOC')
EXECUTE IN_DB_UPGRADE.UPTIME_GET_STATUS
EXECUTE IN_DB_UPGRADE.DOWNTIME_GET_STATUS
```

## Prerequisites

### Database storage for the IN\_DOC table and indexes (EP2 7.5)

The following only applies to upgrades starting on schema versions prior to 7.5.0.0.

To facilitate the process of upgrading the IN\_DOC table in uptime, a copy of the IN\_DOC table is created during the uptime steps and is synchronized with the original table until the downtime steps have completed. This requires that sufficient free space is available to accommodate the copy of the IN\_DOC table and its indexes. Ensure that datafiles are configured to auto extend as needed and that the underlying disks have the capacity to accommodate the expected growth. For best results, manually resize the appropriate datafiles to pre-allocate the additional space to the respective tablespaces to help prevent waits attributed to the allocation of extents during the execution of the uptime steps.

### Additional storage considerations

- Consideration should also be given to the expected volume of new documents created during the time that the synchronization process is in place and additional storage should be allocated to account for that growth.
- During the downtime steps, the original IN\_DOC table and indexes will be replaced by the new versions created during the uptime steps. Upon completion of the upgrade downtime steps and after the original IN\_DOC table and its indexes are dropped the additional storage will be released and become available within the respective tablespaces.
- In a default implementation, the table will be located in the DATA tablespace and all the indexes will be located in the INDX tablespace. You may choose to create the new version of the table and its indexes in a different tablespace if desired.

## Formatting procedure output

The following session formatting is recommended to enhance the readability of the output when using sqlplus to execute the procedures. Sqlplus is the recommended utility for interfacing with the IN\_DB\_UPGRADE package but is not required.

```
set serveroutput on size unlimited format wrapped
set lines 200 pages 1000
set feedback on;
set echo on;
```

## Logging

IN\_DB\_UPGRADE package uses the IN\_DB\_UTIL.IN\_LOGGER procedure to log certain actions to the INUSER.IN\_DB\_UTIL\_LOG table. This table will contain a log of any exceptions that occur during the execution of the IN\_DB\_UTIL and IN\_DB\_UPGRADE packages. It also contains a log of calls to the IN\_SESSION\_SET, and IN\_SESSION\_CLEAR, and IN\_SESSION\_CLEAR\_ALL stored procedures.

To view the contents of the IN\_DB\_UTIL\_LOG table execute the query provided in the [Troubleshooting](#) section.

It is highly recommended that you also utilize the spool command for your session before executing any of the procedures so that you have full logging of the output displayed by the various procedures. Refer to Oracle documentation for utilizing the spool command.

## Create the packages

### IN\_DB\_UTIL package

The IN\_DB\_UTIL package is a collection of integrated procedures and functions that perform common database tasks that are utilized by the IN\_DB\_UPGRADE package. To create the IN\_DB\_UTIL package, complete the following step.

- Run the following script to create the IN\_DB\_UTIL package using a privileged user (SYSDBA).

```
@PerceptiveContentDB_IN_DB_UTIL_Oracle_Package_v4.2.sql
```

### IN\_DB\_UPGRADE package

The IN\_DB\_UPGRADE package is a collection of integrated procedures and functions that manage the synchronization process and execute database schema changes for both uptime and downtime steps. This package is used to upgrade the INOW schema to version 7.9.0.0 or 7.7.0.0 or 7.5.0.0 from schema versions 7.1.3.3 through 7.7.0.0.

- Run the following script to create the IN\_DB\_UPGRADE package using the INUSER user.

```
@PerceptiveContentDB_Upgrade_Package_Oracle_7900.sql
```

## Setup steps

### Create control tables

The UPGRADE\_SETUP procedure is used to create and populate the IN\_DB\_UPGRADE\_CONTROL\_UPGRADE table, which holds the default parameter values that are used if not otherwise specified when executing the upgrade procedures.

The SYNC\_SETUP procedure is used to create the IN\_DB\_UPGRADE\_CONTROL\_SYNC table, which holds the default parameter values that are used if not otherwise specified when executing the synchronization procedures. The SYNC\_SETUP procedure and IN\_DB\_UPGRADE\_CONTROL\_SYNC table only apply to upgrades with a starting schema version less than 7.5.0.0.

#### Upgrade and synchronization (EP2 7.5) default parameter values

For all upgrades

```
TARGET_VERSION      '7.9.0.0'
DEFAULT_PARALLEL    0
DEFAULT_LOGGING     'LOGGING'
```

For upgrades starting with schema versions less than 7.5.0.0

```

DEFAULT_MAX_MINUTES 0
SYNC_BATCH_SIZE      50000
SYNC_BULK_LIMIT      10000
SYNC_VALIDATE_FK     'YES'
SYNC_KEEP_SRC_TABLE  'YES'
SYNC_DATA_TS_NAME    'DATA'
SYNC_INDX_TS_NAME    'INDX'

```

For upgrades starting with schema versions less than 7.2.3.0

```

MSG_DATA_TS_NAME     'IN_MQ_D'
MSG_INDX_TS_NAME     'IN_MQ_I'

```

For all upgrades

```

TABLESPACE_NAME      'DATA'

```

- For any parameters that you do not want to take the default value for, pass the appropriate parameter value when executing `UPGRADE_SETUP` procedure.
- For more details, refer to the [UPGRADE\\_SETUP usage notes and parameters](#) section.
- To create and populate the `IN_DB_UPGRADE_CONTROL_UPGRADE` table, execute the following procedure.

```
EXECUTE INUSER.IN_DB_UPGRADE.UPGRADE_SETUP
```

## Synchronization framework (EP2 7.5)

Applies to upgrades with a starting schema version less than 7.5.0.0.

The synchronization framework supports the propagation of data between the source (`IN_DOC`) and destination (`NEW_DOC`) tables, which facilitate the uptime column modifications and data population and index creation of the new version of the `IN_DOC` table.

**Note** The following steps are executed automatically when `UPGRADE_SETUP` is executed if the schema version is less than 7.5.0.0 and therefore does not need to be manually executed.

- To create and populate the `IN_DB_UPGRADE_CONTROL_SYNC` table, execute the following procedure.

```
EXECUTE IN_DB_UPGRADE.SYNC_SETUP('IN_DOC')
```

- To create the synchronization framework for the `IN_DOC` table, execute the following procedure.

```
EXECUTE IN_DB_UPGRADE.SYNC_SETUP_IN_DOC
```

## Synchronization steps (EP2 7.5)

Applies to upgrades with a starting schema version less than 7.5.0.0.

The following procedures are used to manage the synchronization portion of the upgrade. For more details, refer to the [SYNC\\_TABLE\\_IN\\_DOC usage notes and parameters](#) section. Complete the following steps as needed.

- To synchronize the new version of the `IN_DOC` table (`NEW_DOC`) with the existing version (`IN_DOC`), based on the rows in the staging table (`SYNC_STAGE_DOC`), execute the following procedure. This procedure accepts the `MAX_MINUTES` parameter, among others.

```
EXECUTE INUSER.IN_DB_UPGRADE.SYNC_TABLE_IN_DOC
```

- To view the current status of the synchronization between the source (IN\_DOC) and destination (NEW\_DOC) tables, execute the following procedure.

```
EXECUTE IN_DB_UPGRADE.SYNC_GET_STATUS('IN_DOC')
```

The procedure displays a report on the current state, and history, and performance of the synchronization events.

## Uptime steps (EP2 7.5)

Applies to upgrades with a starting schema version less than 7.5.0.0.

The uptime steps can be executed incrementally over a period of time or all at once depending on the parameters used when executing the UPTIME\_STEPS procedure. For more details, refer to the [UPTIME\\_STEPS usage notes and parameters](#) section. Complete the following steps as needed.

**Note** If you choose not to execute any steps in uptime then you can skip this procedure and go directly to the DOWNTIME\_STEPS procedure.

- To perform qualifying uptime schema changes, execute the following procedure. This procedure accepts the MAX\_MINUTES parameter, among others.

```
EXECUTE INUSER.IN_DB_UPGRADE.UPTIME_STEPS
```

- To view the progress of the qualifying uptime steps, execute the following procedure.

```
EXECUTE INUSER.IN_DB_UPGRADE.UPTIME_GET_STATUS
```

## Downtime steps

### Pre-downtime steps

**Important** Before executing the downtime steps, ensure the following:

- Make sure you have a good full database backup.
- Stop all the services for the Perceptive Content Application server.
- Check for any other connections to the database and disconnect them.
  - You can use the following procedure to check for database connections:

```
EXECUTE INUSER.IN_DB_UTIL.GET_DB_CONNECTIONS
```

### Downtime steps

To execute the qualifying downtime schema changes necessary to upgrade the inuser database schema to 7.7.0.0 or 7.5.0.0, complete the following step.

**Note** The DOWNTIME\_STEPS procedure calls the UPTIME\_STEPS procedure to ensure all qualifying prerequisite actions have completed if the starting schema version is less than 7.5.0.0.

- Execute the following procedure to complete the database downtime steps.

```
spool IN_DB_UPGRADE_DOWNTIME_STEPS.log
```

```
EXECUTE INUSER.IN_DB_UPGRADE.DOWNTIME_STEPS
```

```
spool off
```

- To view the current status and progress of the uptime and downtime steps execute the following procedure from a separate session. Applies to upgrades with a starting schema version less than 7.5.0.0.

```
EXECUTE INUSER.IN_DB_UPGRADE.DOWNTIME_GET_STATUS
```

## Post downtime steps

Upon successful completion of the DOWNTIME\_STEPS, proceed with the following Perceptive Content Server related upgrade steps.

1. Perceptive Content Server related upgrade steps.
2. Full backup of the database.
3. Update database statistics.

```
BEGIN
DBMS_STATS.GATHER_SCHEMA_STATS
(
  OWNNAME           => 'INUSER'
  , ESTIMATE_PERCENT => DBMS_STATS.AUTO_SAMPLE_SIZE
  , METHOD_OPT       => 'FOR ALL COLUMNS SIZE AUTO'
  , CASCADE         => TRUE
  , DEGREE          => DBMS_STATS.AUTO_DEGREE
  , OPTIONS         => 'GATHER AUTO'
);
END;
/
```

4. Conduct standard user acceptance testing procedures.

## View and update upgrade parameter default values

You can query or manually update the IN\_DB\_UPGRADE\_CONTROL\_UPGRADE table to view or change the default values used during the upgrade related procedures.

**Note** You can also update the default upgrade parameters by re-executing the UPGRADE\_SETUP procedure using either the default values or by specifying the desired values for the respective parameters.

- To view the current parameter defaults, execute the following query.

```
SELECT * FROM INUSER.IN_DB_UPGRADE_CONTROL_UPGRADE;
```

- The following is an example of updating default values.

**Note** Not all columns in this table should be modified. Updating some columns can lead to unintended behavior or prevent the package from functioning properly. Ensure to only update the following parameters as needed.

The values used in the following example are not the defaults in some cases. See the parameter descriptions for each procedure in the [Synchronization and Upgrade Procedures](#) section.

```
UPDATE INUSER.IN_DB_UPGRADE_CONTROL_UPGRADE SET
  SCHEMA_TARGET = '7.9.0.0'
  , DEFAULT_MAX_MINUTES = 0
```

```
,DEFAULT_PARALLEL = 4
,DEFAULT_LOGGING = 'LOGGING';

COMMIT;
```

## View and update synchronization parameter default values

Applies to upgrades with a starting schema version less than 7.5.0.0.

You can query or manually update the `IN_DB_UPGRADE_CONTROL_SYNC` table to view or change the default values used during the synchronization related procedures.

**Note** You can also update the default synchronization parameters by re-executing the `SYNC_SETUP` procedure using either the default values or by specifying the desired values for the respective parameters.

- To view the current parameter defaults, execute the following query.

```
SELECT * FROM INUSER.IN_DB_UPGRADE_CONTROL_SYNC;
```

- The following is an example of updating default values.

**Note** Not all columns in this table should be modified. Updating some columns can lead to unintended behavior or prevent the package from functioning properly. Ensure to only update the following parameters as needed.

The values used in the following example are not the defaults in some cases. See the parameter descriptions for each procedure in the [Synchronization and Upgrade Procedures](#) section.

```
UPDATE INUSER.IN_DB_UPGRADE_CONTROL_SYNC SET
  DEFAULT_MAX_MINUTES = 60
, DEFAULT_BATCH_SIZE = 50000
, DEFAULT_BULK_LIMIT = 10000
, DEFAULT_DATA_TS_NAME = 'DATA'
, DEFAULT_INDX_TS_NAME = 'INDX'
, SAVE_TABLE_NAME = 'IN_DOC_SAVE'
, KEEP_SRC_TABLE = 'YES'
, VALIDATE_FK = 'YES'
, SOURCE_TABLE_DEGREE = 1;

COMMIT;
```

## Troubleshooting

- To view active session details, execute the following procedure.

```
EXECUTE INUSER.IN_DB_UTIL.IN_CONTEXT_PRINT
```

- To view a report on uptime progress, execute the following procedure.

```
EXECUTE INUSER.IN_DB_UPGRADE.UPTIME_GET_STATUS
```

- To view a report on downtime progress, execute the following procedure.

```
EXECUTE INUSER.IN_DB_UPGRADE.DOWNTIME_GET_STATUS
```

- To view a report on synchronization events, execute the following procedure.

```
EXECUTE IN_DB_UPGRADE.SYNC_GET_STATUS('IN_DOC')
```

- To view the current state of synchronization, execute the following query. This table is queried by the SYNC\_GET\_STATUS procedure.

```
SELECT * FROM INUSER.SYNC_STATE_DOC;
```

- To view the history of synchronization activity, execute the following query. This table is queried by the SYNC\_GET\_STATUS procedure.

```
SELECT * FROM INUSER.SYNC_HIST_DOC ORDER BY SYNC_DATETIME;
```

- To view the history of synchronization errors, execute the following query. This table is queried by the SYNC\_GET\_STATUS procedure.

```
SELECT * FROM INUSER.SYNC_ERROR_DOC ORDER BY ACTION_DATETIME;
```

- To view the control table for synchronization default values, execute the following query.

```
SELECT * FROM INUSER.IN_DB_UPGRADE_CONTROL_SYNC;
```

- To view the control table for upgrade default values, execute the following query.

```
SELECT * FROM INUSER.IN_DB_UPGRADE_CONTROL_UPGRADE;
```

- The INUSER.IN\_DB\_UTIL\_LOG table contains a log of any exceptions that occur during the execution of the IN\_DB\_UTIL and IN\_DB\_UPGRADE packages. It also contains a log of calls to the IN\_SESSION\_SET, and IN\_SESSION\_CLEAR, and IN\_SESSION\_CLEAR\_ALL stored procedures.

- To view the contents of the IN\_DB\_UTIL\_LOG table execute the following query.

```
SELECT * FROM INUSER.IN_DB_UTIL_LOG ORDER BY LOG_TIME;
```

- For more in-depth troubleshooting see the [Debugging](#) section below to enable the debug option.

## Cleaning up after an early exit

You can use the following procedures to view and manually clear the state for the SYNC or UPGRADE context as needed after an unhandled exception or after manually cancelling an operation before completion. If this happens you will likely encounter one of the following ORA errors the next time you try to execute a procedure within the package. If a procedure terminates in a manner that prevents the post execution cleanup procedures from running then the status for the context will not get reset and prevent new operations from being executed.

If you receive an error similar to the following then a process is either currently executing within another session and it should either be allowed to continue executing or terminated if necessary.

**"ORA-20040: There is another upgrade operation already executing."**

**"ORA-20040: There is another synchronization operation already executing."**

- Execute the following procedure to see what operations are currently running for the SYNC and UPGRADE contexts.

```
EXECUTE INUSER.IN_DB_UTIL.IN_CONTEXT_PRINT
```

If the operation returned by the DISPLAY\_CONTEXT procedure is known to have terminated or been killed then you can execute one of the following procedures to clear the status for both contexts (ALL) or for the specified context (SYNC or UPGRADE).

**IMPORTANT** Do not execute any of the following procedures if any operations are still in progress as it will clear the call-stack and session attributes which will lead to undesired results and unhandled exceptions. If necessary, kill the session first then execute the following to clear the context state.

- To clear the session and call stack for all contexts (SYNC and UPGRADE), execute the following procedure.

```
EXECUTE INUSER.IN_DB_UTIL.IN_SESSION_CLEAR_ALL
```

- To clear session and call stack for only the SYNC context, execute the following procedure.

```
EXECUTE INUSER.IN_DB_UTIL.IN_SESSION_CLEAR_ALL('SYNC')
```

- To clear session and call stack for only the UPGRADE context, execute the following procedure.

```
EXECUTE INUSER.IN_DB_UTIL.IN_SESSION_CLEAR_ALL('UPGRADE')
```

## Upgrade and synchronization procedures

The following procedures provide an interface for users to specify preferences for various parameters used during the upgrade and synchronization steps of the database upgrade. The procedures validate the parameters specified during setup and retain them in control tables for persistence. The default parameter values will be used by the procedures unless different values are specified during the execution of the procedures. The default values can be updated by running the setup procedures again or by manually updating the control tables.

**Note** Synchronization related steps and procedures currently only apply for upgrades with a starting schema version less than 7.5.0.0.

### Upgrade setup procedures

#### UPGRADE\_SETUP

The UPGRADE\_SETUP procedure validates the default parameter values that are used during the upgrade and creates the IN\_DB\_UPGRADE\_CONTROL\_UPGRADE table, which is used to save the default parameter values for the duration of the upgrade. This procedure can be re-executed to update the default upgrade parameter values in the control table or to recreate the IN\_DB\_UPGRADE\_CONTROL\_UPGRADE table if necessary.

If the starting schema version for the upgrade is less than version 7.5.0.0 then the UPGRADE\_SETUP procedure will automatically execute the following synchronization procedures to facilitate the uptime steps for the IN\_DOC table upgrade:

The SYNC\_SETUP procedure, to create and populate the IN\_DB\_UPGRADE\_CONTROL\_SYNC table.

The SYNC\_SETUP\_IN\_DOC procedure, to create the synchronization framework.

#### Procedure definition

```
PROCEDURE UPGRADE_SETUP
(
  TARGET_VERSION          IN VARCHAR2 DEFAULT '7.9.0.0',
  DEFAULT_MAX_MINUTES    IN NUMBER   DEFAULT 0,
  DEFAULT_PARALLEL       IN NUMBER   DEFAULT 0,
  DEFAULT_LOGGING        IN VARCHAR2 DEFAULT 'LOGGING',
  SYNC_BATCH_SIZE        IN NUMBER   DEFAULT 50000,
  SYNC_BULK_LIMIT        IN NUMBER   DEFAULT 10000,
  SYNC_VALIDATE_FK       IN VARCHAR2 DEFAULT 'YES',
  SYNC_KEEP_SRC_TABLE    IN VARCHAR2 DEFAULT 'YES',
  SYNC_DATA_TS_NAME      IN VARCHAR2 DEFAULT NULL,
  SYNC_INDX_TS_NAME      IN VARCHAR2 DEFAULT NULL,
  MSG_DATA_TS_NAME       IN VARCHAR2 DEFAULT NULL,
)
```

```
MSG_INDX_TS_NAME      IN VARCHAR2 DEFAULT NULL,
TABLESPACE_NAME      IN VARCHAR2 DEFAULT NULL
);
```

## Parameters

The UPGRADE\_SETUP procedure accepts the following optional parameters.

| Parameter           | Description   |
|---------------------|---|
| TARGET_VERSION      | The target schema version for the database upgrade. This version of the upgrade package currently supports a target version of 7.5.0.0 and 7.7.0.0 and 7.9.0.0.<br><br>The default value is 7.9.0.0.  |
| DEFAULT_MAX_MINUTES | This only applies for upgrades with a starting schema version less than 7.5.0.0.<br><br>Used to define the default maximum number of minutes that an upgrade or synchronization procedure is permitted to start new operations. When greater than 0, the parameter is used to set an end time that defines when a procedure can no longer pick up new work. This parameter helps to limit the amount of work that a process is permitted to perform to help scope operations within a defined timeframe.<br><br>Applies to table synchronization (SYNC_TABLE_IN_DOC) and non-PK index creation executed by the UPTIME_STEPS procedure.<br><br>The default value of 0 is equal to unlimited. |
| DEFAULT_PARALLEL    | Used to define the default degree of parallelism for qualifying operations. The session will be forced to use the specified degree of parallelism during execution.<br><br>The default value of 0 is equal to CPU_COUNT-2.  |
| DEFAULT_LOGGING     | Used to define the default logging option for qualifying operations. This applies to the creation of tables and indexes. Valid options are LOGGING or NOLOGGING.<br><br>If shipping archived redo logs for replication, then use LOGGING. Otherwise, consider using NOLOGGING to reduce logging during the creation of indexes to improve performance.<br><br>Note that with the NOLOGGING option objects will not be recoverable until they are successfully backed up.<br><br>The default value is LOGGING  |
| SYNC_BATCH_SIZE     | This only applies for upgrades with a starting schema version less than 7.5.0.0.<br><br>Used to define the default batch size for the SYNC_TABLE_IN_DOC procedure. The batch size defines the number of rows to fetch (BULK COLLECT) into the synchronization work queue (cursor) for each batch.<br><br>The default value is 50,000 rows.  |
| SYNC_BULK_LIMIT     | This only applies for upgrades with a starting schema version less than 7.5.0.0.<br><br>Used to define a limit on the number of rows fetched from the cursor by the bulk collect operation during the SYNC_TABLE_IN_DOC procedure. This helps reduce the memory utilization (PGA) for the process.<br><br>The default value is 10,000 rows.   |

|                            |   |
|----------------------------|---|
| <p>SYNC_VALIDATE_FK</p>    | <p>This only applies for upgrades with a starting schema version less than 7.5.0.0.</p> <p>Used to determine if the foreign key constraints, to and from the new IN_DOC table, should be validated after creation. If set to NO, then the SQL to manually validate the FK constraints will be displayed but not executed and must be manually executed as soon as possible after the upgrade.</p> <p>The Foreign Key validation portion of the downtime steps is by far the longest part of the downtime for larger databases. If you are comfortable with the state of the synchronization process heading into the downtime then you can reduce your overall downtime duration by setting this parameter to NO and deferring the FK validation as a manual step after the downtime has completed.</p> <p>The default value of YES means the FK constraints are validated as part of the downtime steps.</p> |
| <p>SYNC_KEEP_SRC_TABLE</p> | <p>This only applies for upgrades with a starting schema version less than 7.5.0.0.</p> <p>Used to specify whether to keep the original/source IN_DOC table upon successful completion of the downtime steps. If YES, then the source table will be renamed and saved after the downtime event. If NO, then the source table will be dropped at the end of the downtime event, which will free up the storage it is consuming.</p> <p>The default value is YES to keep the original table and its PK constraint and index.</p>  |
| <p>SYNC_DATA_TS_NAME</p>   | <p>This only applies for upgrades with a starting schema version less than 7.5.0.0.</p> <p>The name of the tablespace to create the new IN_DOC table in. The user must have a sufficient quota on the specified tablespace.</p> <p>The default value is the current tablespace of the IN_DOC table.</p>   |
| <p>SYNC_INDX_TS_NAME</p>   | <p>This only applies for upgrades with a starting schema version less than 7.5.0.0.</p> <p>The name of the tablespace to create the new IN_DOC table indexes in. The user must have a sufficient quota on the specified tablespace.</p> <p>The default value is MAX(TABLESPACE_NAME) from ALL_INDEXES for the IN_DOC table.</p>   |
| <p>MSG_DATA_TS_NAME</p>    | <p>This only applies for upgrades with a starting schema version less than 7.2.3.0.</p> <p>The name of the tablespace to create the new IN_MESSAGE table in. The user must have a sufficient quota on the specified tablespace.</p> <p>The default value is the current location of the IN_WF_AGENT_STATE table or the IN_MQ_D tablespace, if it exists.</p>  |
| <p>MSG_INDX_TS_NAME</p>    | <p>This only applies for upgrades with a starting schema version less than 7.2.3.0.</p> <p>The name of the tablespace to create the new IN_MESSAGE table indexes in. The user must have a sufficient quota on the specified tablespace.</p> <p>The default value is the current location of the IN_WF_AGENT_STATE primary key index or the IN_MQ_I tablespace, if it exists.</p>  |
| <p>TABLESPACE_NAME</p>     | <p>The name of the tablespace to create the temporary upgrade and synchronization control tables as well as the synchronization framework tables.</p> <p>The default value is the current location of the IN_UPGRADE_TABLE_STATS table.</p>   |

## Examples

```
EXECUTE IN_DB_UPGRADE.UPGRADE_SETUP
EXECUTE IN_DB_UPGRADE.UPGRADE_SETUP('7.7.0.0')
EXECUTE IN_DB_UPGRADE.UPGRADE_SETUP('7.9.0.0', 60, 4, 'LOGGING')

BEGIN
IN_DB_UPGRADE.UPGRADE_SETUP
(
  TARGET_VERSION      => '7.9.0.0'
, DEFAULT_MAX_MINUTES => 0
, DEFAULT_PARALLEL    => 0
, DEFAULT_LOGGING     => 'LOGGING'
, SYNC_BATCH_SIZE     => 50000
, SYNC_BULK_LIMIT     => 10000
, SYNC_VALIDATE_FK    => 'YES'
, SYNC_KEEP_SRC_TABLE => 'YES'
, SYNC_DATA_TS_NAME   => 'DATA'
, SYNC_INDX_TS_NAME   => 'INDX'
, MSG_DATA_TS_NAME    => 'IN_MQ_D'
, MSG_INDX_TS_NAME    => 'IN_MQ_I'
, TABLESPACE_NAME   => 'DATA'
);
END;
/
```

## Synchronization setup procedures (EP2 7.5)

The following synchronization setup procedures are currently only used for upgrades with a starting schema version less than 7.5.0.0.

### SYNC\_SETUP

The SYNC\_SETUP procedure validates the default parameter values used by the synchronization procedures and creates the IN\_DB\_UPGRADE\_CONTROL\_SYNC table, which is used to save the default parameter values for the duration of the upgrade. This procedure can be re-executed manually to update the default synchronization parameter values in the control table or to recreate the IN\_DB\_UPGRADE\_CONTROL\_SYNC table if necessary.

**Note** This procedure is automatically executed by the UPGRADE\_SETUP procedure if the starting schema version is less than 7.5.0.0.

**Note** Manual execution of this procedure should not be necessary unless the SYNC\_REMOVE('IN\_DOC') procedure was manually executed to remove the synchronization framework.

### Procedure definition

```
PROCEDURE SYNC_SETUP
(
  TABLE_NAME          IN VARCHAR2 DEFAULT NULL,
  KEEP_SRC_TABLE       IN VARCHAR2 DEFAULT 'YES',
  DATA_TS_NAME        IN VARCHAR2 DEFAULT NULL,
  INDX_TS_NAME         IN VARCHAR2 DEFAULT NULL,
  MAX_MINUTES          IN NUMBER    DEFAULT 0,
  BATCH_SIZE           IN NUMBER    DEFAULT 50000,
  BULK_LIMIT           IN NUMBER    DEFAULT 10000,
  VALIDATE_FK          IN VARCHAR2 DEFAULT 'YES',
```

```
TABLESPACE_NAME IN VARCHAR2 DEFAULT NULL
);
```

## Parameters

The SYNC\_SETUP procedure accepts the following optional parameters.

| Parameter      | Description   |
|----------------|---|
| TABLE_NAME     | The source table name for the synchronization process. Currently, only the IN_DOC table qualifies for synchronization.  |
| KEEP_SRC_TABLE | Used to specify whether to keep the original/source IN_DOC table upon successful completion of the downtime steps. If YES, then the source table will be renamed and saved after the downtime event. If NO, then the source table will be dropped at the end of the downtime event, which will free up the storage it is consuming.<br><br>The default value is YES to keep the original table and its PK constraint and index.   |
| DATA_TS_NAME   | The name of the tablespace to create the new IN_DOC table in. The user must have a sufficient quota on the specified tablespace.<br><br>The default value is the current tablespace of the IN_DOC table.  |
| INDX_TS_NAME   | The name of the tablespace to create the new IN_DOC table indexes in. The user must have a sufficient quota on the specified tablespace.<br><br>The default value is MAX(TABLESPACE_NAME) from ALL_INDEXES for the IN_DOC table.  |
| MAX_MINUTES    | Used to define the default maximum number of minutes that a synchronization procedure is permitted to start new operations. When greater than 0, the parameter is used to set an end time that defines when a procedure can no longer pick up new work. This parameter helps to limit the amount of work that a process is permitted to perform to help scope operations within a defined timeframe.<br><br>Applies to table synchronization (SYNC_TABLE_IN_DOC) and non-PK index creation executed by the UPTIME_STEPS procedure.<br><br>The default value of 0 is equal to unlimited. |
| BATCH_SIZE     | Used to define the default batch size for the SYNC_TABLE_IN_DOC procedure. The batch size defines the number of rows to fetch (BULK COLLECT) into the synchronization work queue (cursor) for each batch.<br><br>The default value is 50,000 rows.  |
| BULK_LIMIT     | Used to define a limit on the number of rows fetched from the cursor by the bulk collect operation during the SYNC_TABLE_IN_DOC procedure. This helps reduce the memory utilization (PGA) for the process.<br><br>The default value is 10,000 rows.   |
| VALIDATE_FK    | Used to determine if the foreign key constraints, to and from the new IN_DOC table, should be validated after creation. If set to NO, then the SQL to manually validate the FK constraints will be displayed but not executed and must be manually executed as soon as possible after the upgrade.<br><br>The Foreign Key validation portion of the downtime steps is by far the longest part of the downtime for larger databases. If you are comfortable with the state of the  |

|                 |  |
|-----------------|--|
|                 | <p>synchronization process heading into the downtime then you can reduce your overall downtime duration by setting this parameter to NO and deferring the FK validation as a manual step after the downtime has completed.</p> <p>The default value of YES means the FK constraints are validated as part of the downtime steps.</p> |
| TABLESPACE_NAME | <p>The name of the tablespace to create the synchronization control table and synchronization framework tables in.</p> <p>The default value is the current location of the IN_UPGRADE_TABLE_STATS table.</p>   |

### Examples

```
EXECUTE IN_DB_UPGRADE.SYNC_SETUP('IN_DOC')

BEGIN
IN_DB_UPGRADE.SYNC_SETUP
(
TABLE_NAME      => 'IN_DOC'
,KEEP_SRC_TABLE => 'YES'
,DATA_TS_NAME   => 'DATA'
,INDX_TS_NAME   => 'INDX'
,MAX_MINUTES    => 60
,BATCH_SIZE     => 50000
,BULK_LIMIT     => 10000
,VALIDATE_FK    => 'YES'
,TABLESPACE_NAME => 'DATA'
);
END;
/
```

### SYNC\_SETUP\_IN\_DOC

The SYNC\_SETUP\_IN\_DOC procedure creates the synchronization framework that is specific to the IN\_DOC table. This procedure is automatically executed by the UPGRADE\_SETUP procedure if the starting schema version is less than 7.5.0.0. This procedure can be executed manually after running SYNC\_SETUP('IN\_DOC') to recreate the synchronization framework.

**Note** Manual execution of this procedure should not be necessary unless the SYNC\_REMOVE('IN\_DOC') procedure was manually executed to remove the synchronization framework.

### Procedure definition

```
PROCEDURE SYNC_SETUP_IN_DOC
(
TABLESPACE_NAME IN VARCHAR2 DEFAULT NULL
);
```

### Parameters

The SYNC\_SETUP\_IN\_DOC procedure accepts the following optional parameters.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

|                 |   |
|-----------------|---|
| TABLESPACE_NAME | The name of the tablespace to create the synchronization framework tables in.<br>The default value is the current location of the IN_UPGRADE_TABLE_STATS table. |
|-----------------|---|

## Examples

```
EXECUTE IN_DB_UPGRADE.SYNC_SETUP_IN_DOC
EXECUTE IN_DB_UPGRADE.SYNC_SETUP_IN_DOC('DATA')
```

## Synchronization procedures (EP2 7.5)

The following synchronization procedures are currently only used for upgrades with a starting schema version less than 7.5.0.0.

### SYNC\_TABLE\_IN\_DOC

The SYNC\_TABLE\_IN\_DOC procedure synchronizes the source table (IN\_DOC) and the destination table (NEW\_DOC) by propagating changed data that is recorded and staged in the SYNC\_STAGE\_DOC table.

For every row modified (INSERT, UPDATE, DELETE) in the source table (IN\_DOC) the SYNC\_TRG\_DOC trigger will insert a row into the staging table to stage it for synchronization by the SYNC\_TABLE\_IN\_DOC procedure.

During the execution of the SYNC\_TABLE\_IN\_DOC procedure, staged rows are processed in batches according to the BATCH\_SIZE parameter and ordered by the STAGE\_ID to promote a first in first out process. After a batch of rows has been successfully committed, the respective rows are deleted from the staging table. If all staged rows have been successfully processed the SYNC\_STAGE\_DOC table will be empty.

Use the SYNC\_GET\_STATUS procedure to get a detailed report on the current state, and history, and performance of previous executions of the SYNC\_TABLE\_IN\_DOC procedure.

The SYNC\_TABLE\_IN\_DOC procedure accepts the MAX\_MINUTES parameter to limit the amount of work that is completed per execution of the procedure. Using a smaller value for MAX\_MINUTES can help to keep a synchronization event limited to a shorter timeframe per execution of the procedure.

**Note** If document creation volumes are high, the SYNC\_STAGE\_DOC staging table will grow as rows are staged for synchronization. Be sure to execute the SYNC\_TABLE\_IN\_DOC procedure periodically to synchronize the new IN\_DOC table (NEW\_DOC) with the current IN\_DOC table.

**Note** You can schedule this procedure or manually execute it as needed to synchronize rows between the source and destination tables at any point after setup and creation of the destination table and primary key index and prior to the execution of the downtime steps.

### Procedure definition

```
PROCEDURE SYNC_TABLE_IN_DOC
(
    MAX_MINUTES IN NUMBER DEFAULT NULL,
    BATCH_SIZE IN NUMBER DEFAULT NULL,
    BULK_LIMIT IN NUMBER DEFAULT NULL
);
```

### Parameters

The SYNC\_TABLE\_IN\_DOC procedure accepts the following optional parameters.

| Parameter   | Description   |
|-------------|---|
| MAX_MINUTES | <p>The maximum number of minutes that the synchronization process will be allowed to begin processing a new batch of rows.</p> <p>If MAX_MINUTES is greater than 0 then a check will take place to determine if time (MAX_MINUTES) has expired before each new batch of rows is fetched for processing.</p> <p>The default value is defined in the DEFAULT_MAX_MINUTES column of the IN_DB_UPGRADE_CONTROL_SYNC table.</p> <p>The default value of 0 is equal to unlimited which means synchronization will continue until all rows in the SYNC_STAGE_DOC table have been processed and the source and destination tables are fully synchronized.</p> |
| BATCH_SIZE  | <p>The batch size defines the number of rows to fetch (BULK COLLECT) into the synchronization work queue (cursor) for each batch.</p> <p>The default value is defined in the DEFAULT_BATCH_SIZE column of the IN_DB_UPGRADE_CONTROL_SYNC table.</p> <p>The default value is 50,000 rows.</p>  |
| BULK_LIMIT  | <p>The bulk limit on the number of rows fetched from the cursor by the bulk collect operation. This helps reduce the memory utilization (PGA) for the process.</p> <p>The default value is defined in the DEFAULT_BULK_LIMIT column of the IN_DB_UPGRADE_CONTROL_SYNC table.</p> <p>The default value is 10,000 rows.</p>   |

### Examples

```
EXECUTE IN_DB_UPGRADE.SYNC_TABLE_IN_DOC
EXECUTE IN_DB_UPGRADE.SYNC_TABLE_IN_DOC(1, 50000, 10000)
```

### SYNC\_GET\_STATUS

The SYNC\_GET\_STATUS procedure displays the status of synchronization between the source and destination tables. When executed it generates a detailed report on the current state, and history, and performance of previous executions of the SYNC\_TABLE\_IN\_DOC procedure.

#### Procedure definition

```
PROCEDURE SYNC_GET_STATUS
(
  TABLE_NAME IN VARCHAR2 DEFAULT NULL
);
```

#### Parameters

The SYNC\_GET\_STATUS procedure requires the following parameter.

| Parameter  | Description  |
|------------|--|
| TABLE_NAME | The source table name for the synchronization process. |

## Examples

```
EXECUTE IN_DB_UPGRADE.SYNC_GET_STATUS('IN_DOC')
```

## SYNC\_RESET

The SYNC\_RESET procedure resets the synchronization process for the specified table. This includes dropping all indexes (that exist) on the destination table and then truncating all the synchronization framework tables and the destination table then re-populating the destination table from the source table.

This procedure includes the following list of actions but can vary depending on which objects have been created as a result of previous executions of the UPTIME\_STEPS procedure.

- Disable the synchronization trigger
  - Disable SYNC\_TRG\_DOC trigger
- Reset the synchronization sequence
  - Recreate the SYNC\_SEQ\_DOC sequence
- Truncate the synchronization management tables
  - Truncate the SYNC\_STAGE\_DOC table
  - Truncate the SYNC\_ERROR\_DOC table
  - Truncate the SYNC\_STATE\_DOC table
  - Truncate the SYNC\_HIST\_DOC table
- Drop indexes from the NEW\_DOC table if they exist
  - Drop index NEW\_DOC\_IDX10
  - Drop index NEW\_DOC\_IDX11
  - Drop index NEW\_DOC\_IDX12
  - Drop index NEW\_DOC\_IDX2
  - Drop index NEW\_DOC\_IDX5
  - Drop index NEW\_DOC\_IDX6
  - Drop index NEW\_DOC\_IDX7
  - Drop index NEW\_DOC\_IDX81
  - Drop index NEW\_DOC\_IDX82
  - Drop index NEW\_DOC\_IDX83
  - Drop index NEW\_DOC\_IDX84
  - Drop index NEW\_DOC\_IDX85
  - Drop index NEW\_DOC\_IDX9
  - Drop index NEW\_FKCS\_D\_DOC\_TYPE\_ID
  - Drop index NEW\_FKCS\_DOC\_DRAWER\_ID
  - Drop index NEW\_FKCS\_DOC\_INSTANCE\_ID

- Drop foreign key constraints to and from the NEW\_DOC table
- Drop primary key constraint from the NEW\_DOC table
  - Drop Constraint PK\_NEW\_DOC
- Drop primary key index from the INUSER.NEW\_DOC table
  - Drop index PK\_NEW\_DOC
- Drop primary key case Insensitive Index from the INUSER.NEW\_DOC table
  - Drop index PKCI\_NEW\_DOC
- Truncate the NEW\_DOC table
- Re-load of the INUSER.NEW\_DOC table
  - Enable the SYNC\_TRG\_DOC trigger

Re-populate the NEW\_DOC table from the source table

- Create primary key on the INUSER.NEW\_DOC table
  - Create primary key constraint on the NEW\_DOC table as disabled
  - Create primary key index PK\_NEW\_DOC on the NEW\_DOC table
  - Enable and validate the PK\_NEW\_DOC primary key on the NEW\_DOC table
  - Create the case insensitive primary key index PKCI\_NEW\_DOC on the NEW\_DOC table

### Procedure definition

```
PROCEDURE SYNC_RESET
(
  TABLE_NAME IN VARCHAR2 DEFAULT NULL,
  PARALLEL    IN VARCHAR2 DEFAULT NULL,
  LOGGING     IN NUMBER    DEFAULT NULL
);
```

### Parameters

The SYNC\_RESET procedure accepts the following optional parameters.

| Parameter  | Description  |
|------------|--|
| TABLE_NAME | The source table name for which to reset the synchronization framework.  |
| PARALLEL   | The degree of parallelism for the session during re-population and primary key index creation on the destination table.<br><br>The default value is defined in the DEFAULT_PARALLEL column of the IN_DB_UPGRADE_CONTROL_UPGRADE table. |

|         |  |
|---------|--|
| LOGGING | <p>Used to define the default logging option for qualifying operations. This applies to the creation of tables and indexes. Valid options are LOGGING or NOLOGGING.</p> <p>If shipping archived redo logs for replication, then use LOGGING. Otherwise, consider using NOLOGGING to reduce logging during the creation of indexes to improve performance.</p> <p>Note that with the NOLOGGING option, these objects will not be recoverable until they are successfully backed up.</p> <p>The default value is defined in the DEFAULT_LOGGING column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> |
|---------|--|

## Examples

```
EXECUTE IN_DB_UPGRADE.SYNC_RESET('IN_DOC')
EXECUTE IN_DB_UPGRADE.SYNC_RESET('IN_DOC', 0, 'NOLOGGING')
```

## SYNC\_REMOVE

The SYNC\_REMOVE procedure removes the synchronization framework and upgrade objects including all of the following if they exist.

- NEW\_DOC table and indexes. (Default Destination Table for pre 7.5.0.0 upgrades)
- SYNC\_TRG\_DOC trigger
- SYNC\_SEQ\_DOC sequence
- SYNC\_ERROR\_DOC table
- SYNC\_STATE\_DOC table
- SYNC\_HIST\_DOC table
- SYNC\_STAGE\_DOC table

## Procedure definition

```
PROCEDURE SYNC_REMOVE
(
  TABLE_NAME IN VARCHAR2 DEFAULT NULL
);
```

## Parameter

The SYNC\_REMOVE procedure requires the following parameter.

| Parameter  | Description   |
|------------|---|
| TABLE_NAME | The source table name for which to reset the synchronization framework. |

## Example

```
EXECUTE IN_DB_UPGRADE.SYNC_REMOVE('IN_DOC')
```

## SYNC\_CHECK

The SYNC\_CHECK procedure can be used to check the synchronization framework for the specified table. If any objects are missing it will display instructions to repair the synchronization framework.

### Procedure definition

```
PROCEDURE SYNC_CHECK
(
  TABLE_NAME          IN VARCHAR2 DEFAULT NULL,
  CHECK_DEST_TABLE     IN VARCHAR2 DEFAULT 'YES'
);
```

### Parameters

The SYNC\_CHECK procedure accepts the following parameters.

| Parameter        | Description   |
|------------------|---|
| TABLE_NAME       | The source table name for which to check the synchronization framework. Required.                 |
| CHECK_DEST_TABLE | Indicates if you want to check for the existence of the new/destination table.<br>Default is YES. |

### Examples

```
EXECUTE IN_DB_UPGRADE.SYNC_CHECK('IN_DOC')
EXECUTE IN_DB_UPGRADE.SYNC_CHECK('IN_DOC', 'YES')
EXECUTE IN_DB_UPGRADE.SYNC_CHECK('IN_DOC', 'NO')
```

## Upgrade procedures

The following applies to upgrades with a starting schema version of 7.5.0.0 and up. For upgrades starting with a schema version less than 7.5.0.0, refer to the [Upgrade Procedures \(EP2 7.5\)](#) section.

### DOWNTIME\_STEPS

The DOWNTIME\_STEPS procedure calls all the other required procedures, in the order they should be executed, to upgrade the inuser schema.

This procedure executes the following procedures in the following order:

```
IN_DB_UTIL.GET_DB_CONNECTIONS, IN_DB_UTIL.GET_DB_CONNECTIONS_COUNT,
SCHEMA_UPGRADE_7500_7700, SCHEMA_UPGRADE_7700_7900,
IN_DB_UTIL.PRINT_SCHEMA_HIST.
```

### Procedure definition

```
PROCEDURE DOWNTIME_STEPS
(
  PARALLEL           IN NUMBER   DEFAULT NULL,
  LOGGING            IN VARCHAR2 DEFAULT NULL,
  MAX_CONNECTIONS    IN NUMBER   DEFAULT 5
);
```

## Parameters

The UPTIME\_STEPS procedure accepts the following optional parameters.

| Parameter       | Description   |
|-----------------|---|
| PARALLEL        | <p>The degree of parallelism for the session.</p> <p>The default value is defined in the DEFAULT_PARALLEL column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value of 0 is equal to CPU_COUNT-2.</p>   |
| LOGGING         | <p>Used to define the default logging option for qualifying operations. This applies to the creation of tables and indexes. Valid options are LOGGING or NOLOGGING.</p> <p>If shipping archived redo logs for replication, then use LOGGING. Otherwise, consider using NOLOGGING to reduce logging during the creation of indexes to improve performance.</p> <p>Note that with the NOLOGGING option, these objects will not be recoverable until they are successfully backed up.</p> <p>The default value is defined in the DEFAULT_LOGGING column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value is LOGGING.</p> |
| MAX_CONNECTIONS | <p>Used to specify how many DB connections to allow during the downtime connections checks. If the number of connections exceed this number, then the upgrade will not begin.</p> <p>This check is based on the results of the following function:</p> <pre>SELECT IN_DB_UTIL.GET_DB_CONNECTIONS_COUNT FROM DUAL;</pre> <p>The default value is 5.</p>  |

## Examples

```
EXECUTE IN_DB_UPGRADE.DOWNTIME_STEPS
EXECUTE IN_DB_UPGRADE.DOWNTIME_STEPS (0, 'LOGGING', 5)
```

## Upgrade procedures (EP2 7.5)

The following procedures are currently only used for upgrades with a starting schema version less than 7.5.0.0.

### UPTIME\_STEPS

The UPTIME\_STEPS procedure will execute all the necessary procedures that contain qualifying uptime steps, in the order they should be executed. To minimize the impact of executing the uptime steps, it is recommended to execute this procedure during periods of low activity.

This procedure accepts the MAX\_MINUTES parameter to impose a limit on the amount of work that is completed per execution of the procedure.

To further reduce the impact to other users, consider using smaller values for the degree of parallelism (PARALLEL) to limit the number of processors that the procedure can use. Note that using less processors will result in the operation taking longer to complete.

Alternatively, using a higher degree of parallelism and a smaller value for MAX\_MINUTES can help to complete a single task as quickly as possible.

The use of these parameters can help to facilitate the incremental creation of the required objects during uptime in the days or hours leading up to the downtime with minimal impact to the rest of the system.

**Note** You may also consider leveraging Oracle consumer groups and map the UPGRADE or SYNC client identifiers (CLIENT\_ID) to a specific consumer group to restrict the resources that an upgrade or synchronization procedure may utilize. Beware that doing so may result in a slower upgrade or downtime event if the upgrade processes are constrained and unable to run freely.

**Note** You can schedule this procedure or manually execute it as necessary to incrementally execute the uptime steps at any point after upgrade setup and prior to the execution of the downtime steps.

This procedure executes the following procedures in the following order:  
 SCHEMA\_UPGRADE\_7220\_7230, SCHEMA\_UPGRADE\_7403\_7500, UPTIME\_STEPS\_IN\_DOC

### Procedure definition

```
PROCEDURE UPTIME_STEPS
(
  MAX_MINUTES IN NUMBER DEFAULT NULL,
  PARALLEL    IN NUMBER DEFAULT NULL,
  LOGGING     IN VARCHAR2 DEFAULT NULL
);
```

### Parameters

The UPTIME\_STEPS procedure accepts the following optional parameters.

| Parameter   | Description  |
|-------------|--|
| MAX_MINUTES | <p>The maximum number of minutes that the procedure is permitted to start new operations. When greater than 0, the parameter is used to set an end time that defines when the procedure can no longer pick up new work. This parameter helps to limit the amount of work that a procedure is permitted to perform to help scope operations within a limited timeframe.</p> <p>Applies to non-PK index creation executed by the UPTIME_STEPS procedure.</p> <p>The default value is defined in the DEFAULT_MAX_MINUTES column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value of 0 is equal to unlimited which means uptime steps will continue running until all steps have been completed.</p> |
| PARALLEL    | <p>The degree of parallelism to use for the session.</p> <p>The default value is defined in the DEFAULT_PARALLEL column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value of 0 is equal to CPU_COUNT-2.</p>   |

|         |   |
|---------|---|
| LOGGING | <p>Used to define the default logging option for qualifying operations. This applies to the creation of tables and indexes. Valid options are LOGGING or NOLOGGING.</p> <p>If shipping archived redo logs for replication, then use LOGGING. Otherwise, consider using NOLOGGING to reduce logging during the creation of indexes to improve performance.</p> <p>Note that with the NOLOGGING option, these objects will not be recoverable until they are successfully backed up.</p> <p>The default value is defined in the DEFAULT_LOGGING column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value is LOGGING.</p> |
|---------|---|

## Examples

```
EXECUTE IN_DB_UPGRADE.UPTIME_STEPS
EXECUTE IN_DB_UPGRADE.UPTIME_STEPS (60, 10, 'LOGGING')
```

## UPTIME\_GET\_STATUS

The UPTIME\_GET\_STATUS procedure displays the progress of the upgrade uptime steps.

### Procedure definition

```
PROCEDURE UPTIME_GET_STATUS;
```

### Parameters

The UPTIME\_GET\_STATUS procedure does not accept any parameters.

### Example

```
EXECUTE IN_DB_UPGRADE.UPTIME_GET_STATUS
```

## DOWNTIME\_STEPS

The DOWNTIME\_STEPS procedure applies to all upgrades, but the following summary applies to upgrades with a starting schema version less than 7.5.0.0. The DOWNTIME\_STEPS procedure calls all the other uptime and downtime related procedures in the order they should be executed.

This procedure executes the following procedures in the following order:

```
IN_DB_UTIL.GET_DB_CONNECTIONS, IN_DB_UTIL.GET_DB_CONNECTIONS_COUNT,
UPTIME_STEPS, SYNC_GET_STATUS, SYNC_DISABLE_TRIGGER, SYNC_TABLE_IN_DOC,
CREATE_FK_IN_DOC, IN_DB_UTIL.DROP_FK, IN_DB_UTIL.DROP_INDEX,
SCHEMA_UPGRADE_7133_7140, SCHEMA_UPGRADE_7140_7152,
SCHEMA_UPGRADE_7152_7201, SCHEMA_UPGRADE_7220_7230,
SCHEMA_UPGRADE_7230_7403, SCHEMA_UPGRADE_7403_7500,
SCHEMA_UPGRADE_7500_7700, SCHEMA_UPGRADE_7700_7900, SYNC_REMOVE,
IN_DB_UTIL.PRINT_SCHEMA_HIST.
```

**NOTE** When DOWNTIME\_STEPS is executed, the MAX\_MINUTES parameter is hard coded to 0 which is equal to unlimited so that it runs to completion.

**NOTICE** Oracle 11.2.0.4 - Bug 28855186 : PARALLELISM NOT WORKING WHEN TRYING TO ENABLE VALIDATE CONSTRAINT on Oracle 11.2.0.4.180717 and up. This Oracle bug prevents the parallel validation of foreign key constraints for the affected versions of Oracle.

### Procedure definition

```
PROCEDURE DOWNTIME_STEPS
(
  PARALLEL          IN NUMBER    DEFAULT NULL,
  LOGGING           IN VARCHAR2  DEFAULT NULL,
  MAX_CONNECTIONS  IN NUMBER    DEFAULT 5
);
```

### Parameters

The UPTIME\_STEPS procedure accepts the following optional parameters

| Parameter       | Description   |
|-----------------|---|
| PARALLEL        | <p>The degree of parallelism to use for the session.</p> <p>The default value is defined in the DEFAULT_PARALLEL column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value of 0 is equal to CPU_COUNT-2.</p>  |
| LOGGING         | <p>Used to define the default logging option for qualifying operations. This applies to the creation of tables and indexes. Valid options are LOGGING or NOLOGGING.</p> <p>If shipping archived redo logs for replication, then use LOGGING. Otherwise, consider using NOLOGGING to reduce logging during the creation of indexes to improve performance.</p> <p>Note that with the NOLOGGING option, these objects will not be recoverable until they are successfully backed up.</p> <p>The default value is defined in the DEFAULT_LOGGING column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value is LOGGING.</p> |
| MAX_CONNECTIONS | <p>Used to specify how many DB connections to allow during the downtime connections checks. If the number of connections exceed this number, then the upgrade will not begin.</p> <p>This check is based on the results of the following function:</p> <pre>SELECT IN_DB_UTIL.GET_DB_CONNECTIONS_COUNT FROM DUAL;</pre> <p>The default value is 5.</p>  |

### Examples

```
EXECUTE IN_DB_UPGRADE.DOWNTIME_STEPS
EXECUTE IN_DB_UPGRADE.DOWNTIME_STEPS (0, 'LOGGING', 5)
```

## DOWNTIME\_GET\_STATUS

The DOWNTIME\_GET\_STATUS procedure displays the progress of the upgrade downtime and uptime steps.

### Procedure definition

```
PROCEDURE DOWNTIME_GET_STATUS;
```

### Parameters

The DOWNTIME\_GET\_STATUS procedure does not accept any parameters.

### Example

```
EXECUTE IN_DB_UPGRADE.DOWNTIME_GET_STATUS
```

## CREATE\_DEST\_TABLE\_IN\_DOC

The CREATE\_DEST\_TABLE\_IN\_DOC procedure creates the destination table used by the synchronization process for the IN\_DOC table. This version of the IN\_DOC table represents its structure as of schema version 7.5.0.0 and up. This procedure is executed indirectly by the UPTIME\_STEPS procedure or can be executed manually after running SYNC\_SETUP\_IN\_DOC.

### Procedure definition

```
PROCEDURE CREATE_DEST_TABLE_IN_DOC
(
  PARALLEL          IN NUMBER    DEFAULT NULL,
  LOGGING           IN VARCHAR2  DEFAULT NULL,
  TABLESPACE_NAME IN VARCHAR2  DEFAULT NULL
);
```

### Parameters

The CREATE\_DEST\_TABLE\_IN\_DOC procedure accepts the following optional parameters.

| Parameter | Description   |
|-----------|---|
| PARALLEL  | <p>The degree of parallelism to use for the session. This parameter is used during the creation of the destination table (DEGREE parameter) and will dictate the number of threads per instance for future scans on the table. During the downtime steps the final DEGREE parameter value for the table will be set to the current value of the IN_DOC table as recorded in the SOURCE_TABLE_DEGREE column of the IN_DB_UPGRADE_CONTROL_SYNC table.</p> <p>The default value is defined in the DEFAULT_PARALLEL column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value of 0 is equal to CPU_COUNT-2.</p> |

|                 |  |
|-----------------|--|
| LOGGING         | <p>Used to define whether the creation of the destination table is logged (LOGGING) or not logged (NOLOGGING). If shipping archived redo logs for replication, then use LOGGING.</p> <p>Note that with the NOLOGGING option the table will not be recoverable until successfully backed up.</p> <p>The default value is defined in the DEFAULT_LOGGING column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value is LOGGING.</p> |
| TABLESPACE_NAME | <p>The name of the tablespace to create the destination table in. The user must have a sufficient quota on the tablespace.</p> <p>The default value is defined in the DEFAULT_DATA_TS_NAME column of the IN_DB_UPGRADE_CONTROL_SYNC table.</p> <p>The default value is the current location of the IN_DOC table.</p>   |

### Examples

```
EXECUTE IN_DB_UPGRADE.CREATE_DEST_TABLE_IN_DOC
EXECUTE IN_DB_UPGRADE.CREATE_DEST_TABLE_IN_DOC (0, 'LOGGING', 'DATA')
```

## LOAD\_DEST\_TABLE\_IN\_DOC

The `LOAD_DEST_TABLE_IN_DOC` procedure executes a direct-path copy of rows from the source table (`IN_DOC`) to the destination table (`NEW_DOC`). This procedure is indirectly executed by the `UPTIME_STEPS` procedure or can be executed manually after running `CREATE_DEST_TABLE_IN_DOC`.

### Procedure definition

```
PROCEDURE LOAD_DEST_TABLE_IN_DOC
(
  PARALLEL IN NUMBER DEFAULT NULL
);
```

### Parameters

The `LOAD_DEST_TABLE_IN_DOC` procedure accepts the following optional parameters.

| Parameter | Description   |
|-----------|---|
| PARALLEL  | <p>The degree of parallelism for the direct-path copy of data between the source table (<code>IN_DOC</code>) to the destination table (<code>NEW_DOC</code>).</p> <p>The default value is defined in the DEFAULT_PARALLEL column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value of 0 is equal to CPU_COUNT-2.</p> |

### Examples

```
EXECUTE IN_DB_UPGRADE.LOAD_DEST_TABLE_IN_DOC
EXECUTE IN_DB_UPGRADE.LOAD_DEST_TABLE_IN_DOC (0)
```

## CREATE\_PK\_IN\_DOC

The CREATE\_PK\_IN\_DOC procedure creates the primary key constraint and primary key index and the primary key case insensitive (PKCI) NLS function-based index on the new IN\_DOC table (NEW\_DOC). This procedure is indirectly executed by the UPTIME\_STEPS procedure or can be executed manually after running CREATE\_DEST\_TABLE\_IN\_DOC, and LOAD\_DEST\_TABLE\_IN\_DOC.

**Note** The indexes created by this procedure must be in place prior to starting the synchronization process or the performance of the sync process will be severely impacted.

**Note** This procedure will create both the primary key index (PK\_NEW\_DOC) and the case insensitive unique index (PKCI\_NEW\_DOC) and does not honor the MAX\_MINUTES parameter.

### Procedure definition

```
PROCEDURE CREATE_PK_IN_DOC
(
  PARALLEL          IN NUMBER    DEFAULT NULL,
  LOGGING           IN VARCHAR2  DEFAULT NULL,
  TABLESPACE_NAME IN VARCHAR2  DEFAULT NULL
);
```

### Parameters

The CREATE\_PK\_IN\_DOC procedure accepts the following optional parameters.

| Parameter       | Description  |
|-----------------|--|
| PARALLEL        | <p>The degree of parallelism to use while creating the primary key constraint and indexes on the destination table.</p> <p>The default value is defined in the DEFAULT_PARALLEL column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value of 0 is equal to CPU_COUNT-2.</p>  |
| LOGGING         | <p>Used to define whether the index operations are logged (LOGGING) or not logged (NOLOGGING). If shipping archived redo logs for replication, then use LOGGING otherwise NOLOGGING will improve performance.</p> <p>Note that with the NOLOGGING option, the indexes will not be recoverable until successfully backed up.</p> <p>The default value is defined in the DEFAULT_LOGGING column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value is LOGGING.</p> |
| TABLESPACE_NAME | <p>The name of the tablespace to create the destination table indexes in. The user must have a sufficient quota on the tablespace.</p> <p>If not specified, the default value is based on querying MAX(TABLESPACE_NAME) from ALL_INDEXES for the source table.</p> <p>The default value is defined in the DEFAULT_INDX_TS_NAME column of the IN_DB_UPGRADE_CONTROL_SYNC table.</p>   |

## Examples

```
EXECUTE IN_DB_UPGRADE.CREATE_PK_IN_DOC
EXECUTE IN_DB_UPGRADE.CREATE_PK_IN_DOC (0, 'LOGGING', 'INDX')
```

## CREATE\_INDEXES\_IN\_DOC

The CREATE\_INDEXES\_IN\_DOC procedure creates all the non-primary key indexes on the new IN\_DOC table (NEW\_DOC). This procedure is indirectly executed by the UPTIME\_STEPS procedure or can be executed manually after running CREATE\_DEST\_TABLE\_IN\_DOC, and LOAD\_DEST\_TABLE\_IN\_DOC, and CREATE\_PK\_IN\_DOC.

This procedure accepts the MAX\_MINUTES parameter to limit the amount of work that is completed per execution of the procedure.

To further reduce the impact to other users, consider using smaller values for the degree of parallelism (PARALLEL) to limit the number of processors that the procedure can use. Note that using less processors will result in the operation taking longer to complete.

Alternatively, using a higher degree of parallelism and a smaller value for MAX\_MINUTES can help to complete a single task as quickly as possible.

The use of these parameters can help to facilitate the incremental creation of the required indexes during uptime in the days or hours leading up to the downtime with minimal impact to the rest of the system.

**Note** You may also consider leveraging Oracle consumer groups and map the UPGRADE client identifier (CLIENT\_ID) to a specific consumer group to restrict the resources that an upgrade or synchronization procedure may utilize. Beware that doing so may result in a slower upgrade or downtime event if the upgrade processes are constrained and unable to run freely.

**Note** You can schedule this procedure or manually execute it as necessary to incrementally create the indexes prior to the execution of the downtime steps.

## Procedure definition

```
PROCEDURE CREATE_INDEXES_IN_DOC
(
  MAX_MINUTES      IN NUMBER      DEFAULT NULL,
  PARALLEL         IN NUMBER      DEFAULT NULL,
  LOGGING          IN VARCHAR2    DEFAULT NULL,
  TABLESPACE_NAME IN VARCHAR2    DEFAULT NULL
);
```

## Parameters

The CREATE\_INDEXES\_IN\_DOC procedure accepts the following optional parameters.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

|                        |   |
|------------------------|---|
| <p>MAX_MINUTES</p>     | <p>The maximum number of minutes that the procedure is permitted to start new operations. When greater than 0, the parameter is used to set an end time that defines when the procedure can no longer pick up new work. This parameter helps to limit the amount of work that the procedure is permitted to perform to help scope operations within a limited timeframe.</p> <p>If MAX_MINUTES is greater than 0 then a check will take place to determine if time (MAX_MINUTES) has expired before creating the next index on the list.</p> <p>The default value is defined in the DEFAULT_MAX_MINUTES column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value of 0 is equal to unlimited which means the procedure will continue running until all the indexes have been created.</p> |
| <p>PARALLEL</p>        | <p>The degree of parallelism to use while creating the indexes on the destination table.</p> <p>The default value is defined in the DEFAULT_PARALLEL column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value of 0 is equal to CPU_COUNT-2.</p>  |
| <p>LOGGING</p>         | <p>Used to define whether the index operations are logged (LOGGING) or not logged (NOLOGGING). If shipping archived redo logs for replication, then use LOGGING otherwise NOLOGGING will improve performance.</p> <p>Note that with the NOLOGGING option, the indexes will not be recoverable until successfully backed up.</p> <p>The default value is defined in the DEFAULT_LOGGING column of the IN_DB_UPGRADE_CONTROL_UPGRADE table.</p> <p>The default value is LOGGING.</p>  |
| <p>TABLESPACE_NAME</p> | <p>The name of the tablespace to create the indexes in. The user must have a sufficient quota on the tablespace.</p> <p>If not specified, the default value is based on querying MAX(TABLESPACE_NAME) from ALL_INDEXES for the IN_DOC table.</p> <p>The default value is defined in the DEFAULT_INDX_TS_NAME column of the IN_DB_UPGRADE_CONTROL_SYNC table.</p>  |

### Examples

```
EXECUTE IN_DB_UPGRADE.CREATE_INDEXES_IN_DOC
EXECUTE IN_DB_UPGRADE.CREATE_INDEXES_IN_DOC (0, 0, 'LOGGING', 'INDX')
```

## Upgrade objects

### IN\_DB\_UPGRADE\_CONTROL\_UPGRADE table

This table contains the default parameter values that an upgrade procedure will use if not otherwise specified when a procedure is executed.

This table is created and populated during UPGRADE\_SETUP and can be manually updated as needed to adjust certain run-time parameters.

| ❖ COLUMN_ID | ❖ COLUMN_NAME       | ❖ DATA_TYPE        | ❖ NULLABLE | ❖ DATA_DEFAULT |
|-------------|---------------------|--------------------|------------|----------------|
| 1           | DATABASE_NAME       | VARCHAR2(9 BYTE)   | No         | (null)         |
| 2           | SCHEMA_NAME         | VARCHAR2(128 BYTE) | No         | (null)         |
| 3           | SCHEMA_VERSION      | VARCHAR2(40 BYTE)  | No         | (null)         |
| 4           | SCHEMA_TARGET       | VARCHAR2(40 BYTE)  | No         | (null)         |
| 5           | DEFAULT_MAX_MINUTES | NUMBER             | No         | (null)         |
| 6           | DEFAULT_PARALLEL    | NUMBER             | No         | (null)         |
| 7           | DEFAULT_LOGGING     | VARCHAR2(9 BYTE)   | No         | (null)         |
| 8           | IS_SYNC_UPGRADE     | VARCHAR2(3 BYTE)   | No         | (null)         |
| 9           | IDENTIFIER_UPGRADE  | VARCHAR2(128 BYTE) | No         | (null)         |
| 10          | IDENTIFIER_SYNC     | VARCHAR2(128 BYTE) | No         | (null)         |

**Primary Key:** PK\_IN\_DB\_UPGRADE\_CONTROL\_UPGRADE (DATABASE\_NAME, SCHEMA\_NAME)

## Synchronization objects (EP2 7.5)

The following procedures are currently only used for upgrades with a starting schema version less than 7.5.0.0.

### IN\_DB\_UPGRADE\_CONTROL\_SYNC table

This table contains the default parameter values that a synchronization procedure will use if not otherwise specified when a procedure is executed.

This table is created and populated during SYNC\_SETUP and can be manually updated as needed to adjust certain run-time parameters.

| COLUMN_ID | COLUMN_NAME             | DATA_TYPE          | NULLABLE | DATA_DEFAULT |
|-----------|-------------------------|--------------------|----------|--------------|
| 1         | DATABASE_NAME           | VARCHAR2(30 BYTE)  | No       | (null)       |
| 2         | SCHEMA_NAME             | VARCHAR2(128 BYTE) | No       | (null)       |
| 3         | SOURCE_TABLE_NAME       | VARCHAR2(128 BYTE) | No       | (null)       |
| 4         | BASE_TABLE_NAME         | VARCHAR2(128 BYTE) | No       | (null)       |
| 5         | DEST_TABLE_NAME         | VARCHAR2(128 BYTE) | No       | (null)       |
| 6         | SAVE_TABLE_NAME         | VARCHAR2(128 BYTE) | No       | (null)       |
| 7         | DEFAULT_MAX_MINUTES     | NUMBER             | No       | (null)       |
| 8         | DEFAULT_BATCH_SIZE      | NUMBER             | No       | (null)       |
| 9         | DEFAULT_BULK_LIMIT      | NUMBER             | No       | (null)       |
| 10        | DEFAULT_DATA_TS_NAME    | VARCHAR2(128 BYTE) | No       | (null)       |
| 11        | DEFAULT_INDX_TS_NAME    | VARCHAR2(128 BYTE) | No       | (null)       |
| 12        | STAGE_TABLE_NAME        | VARCHAR2(128 BYTE) | No       | (null)       |
| 13        | ERROR_TABLE_NAME        | VARCHAR2(128 BYTE) | No       | (null)       |
| 14        | STATE_TABLE_NAME        | VARCHAR2(128 BYTE) | No       | (null)       |
| 15        | HIST_TABLE_NAME         | VARCHAR2(128 BYTE) | No       | (null)       |
| 16        | SEQUENCE_NAME           | VARCHAR2(128 BYTE) | No       | (null)       |
| 17        | TRIGGER_NAME            | VARCHAR2(128 BYTE) | No       | (null)       |
| 18        | KEEP_SRC_TABLE          | VARCHAR2(3 BYTE)   | No       | (null)       |
| 19        | VALIDATE_FK             | VARCHAR2(3 BYTE)   | No       | (null)       |
| 20        | SOURCE_TABLE_DEGREE     | VARCHAR2(10 BYTE)  | No       | (null)       |
| 21        | CONTEXT_IDENTIFIER_SYNC | VARCHAR2(128 BYTE) | No       | (null)       |

**Primary Key:** PK\_IN\_DB\_UPGRADE\_CONTROL\_SYNC (DATABASE\_NAME, SCHEMA\_NAME, SOURCE\_TABLE\_NAME)

## SYNC\_ERROR\_DOC table

This table contains any errors that are encountered during execution of the SYNC\_TABLE\_IN\_DOC procedure.

| COLUMN_ID | COLUMN_NAME     | DATA_TYPE          | NULLABLE | DATA_DEFAULT |
|-----------|-----------------|--------------------|----------|--------------|
| 1         | STAGE_ID        | NUMBER             | No       | (null)       |
| 2         | ACTION          | VARCHAR2(6 BYTE)   | Yes      | (null)       |
| 3         | ACTION_DATETIME | TIMESTAMP(6)       | No       | (null)       |
| 4         | SRC_PK_COLUMN1  | VARCHAR2(23 BYTE)  | Yes      | (null)       |
| 5         | ERROR_MESSAGE   | VARCHAR2(500 BYTE) | Yes      | (null)       |

**Primary Key:** PK\_SYNC\_ERROR\_DOC (STAGE\_ID, ACTION\_DATETIME)

## SYNC\_HIST\_DOC table

This table contains historical synchronization event details for each execution of the SYNC\_TABLE\_IN\_DOC procedure including performance and throughput metrics and run-time parameters.

| COLUMN_ID | COLUMN_NAME     | DATA_TYPE         | NULLABLE | DATA_DEFAULT |
|-----------|-----------------|-------------------|----------|--------------|
| 1         | SYNC_DATETIME   | TIMESTAMP(6)      | No       | (null)       |
| 2         | SYNC_RPM        | NUMBER            | Yes      | (null)       |
| 3         | SYNC_ROWS       | NUMBER            | Yes      | (null)       |
| 4         | SYNC_SECS       | NUMBER            | Yes      | (null)       |
| 5         | SYNC_BATCH_SIZE | NUMBER            | Yes      | (null)       |
| 6         | SYNC_BULK_LIMIT | NUMBER            | Yes      | (null)       |
| 7         | SYNC_MAX_MINS   | NUMBER            | Yes      | (null)       |
| 8         | SYNC_RESULT     | VARCHAR2(10 BYTE) | Yes      | (null)       |

**Primary Key:** PK\_SYNC\_HIST\_DOC (SYNC\_DATETIME)

## SYNC\_STAGE\_DOC table

This table holds information about modified rows in the source table (IN\_DOC).

For every row modified (INSERT, UPDATE, DELETE) in the source table (IN\_DOC) the SYNC\_TRG\_DOC trigger will insert a row into the staging table to stage it for synchronization by the SYNC\_TABLE\_IN\_DOC procedure.

If document creation volumes are high, this staging table will also grow. Be sure to execute the SYNC\_TABLE\_IN\_DOC procedure periodically to synchronize the new IN\_DOC table (NEW\_DOC) with the current IN\_DOC table.

During the execution of the SYNC\_TABLE\_IN\_DOC procedure, to synchronize data between the source and destination tables, rows are processed in batches according to the batch size and ordered by the STAGE\_ID to promote a first in first out process. After a batch of rows has been successfully processed and committed then the rows will be removed from the staging table.

| COLUMN_ID | COLUMN_NAME       | DATA_TYPE         | NULLABLE | DATA_DEFAULT |
|-----------|-------------------|-------------------|----------|--------------|
| 1         | STAGE_ID          | NUMBER            | No       | (null)       |
| 2         | SRC_PK_COLUMN1    | VARCHAR2(23 BYTE) | No       | (null)       |
| 3         | SRC_CREATION_TIME | TIMESTAMP(6)      | No       | (null)       |
| 4         | ACTION            | VARCHAR2(6 BYTE)  | No       | (null)       |
| 5         | ACTION_DATETIME   | TIMESTAMP(6)      | No       | (null)       |

**Primary Key:** PK\_SYNC\_STAGE\_DOC (STAGE\_ID)

## SYNC\_STATE\_DOC table

This table contains details for the last synchronization event (execution of SYNC\_TABLE\_IN\_DOC).

| ❖ COLUMN_ID | ❖ COLUMN_NAME        | ❖ DATA_TYPE        | ❖ NULLABLE | ❖ DATA_DEFAULT |
|-------------|----------------------|--------------------|------------|----------------|
| 1           | SOURCE_TABLE_NAME    | VARCHAR2(128 BYTE) | No         | (null)         |
| 2           | DEST_TABLE_NAME      | VARCHAR2(128 BYTE) | No         | (null)         |
| 3           | TOTAL_ROWS_SYNCED    | NUMBER             | No         | 0              |
| 4           | LAST_SYNC_DATETIME   | TIMESTAMP(6)       | Yes        | (null)         |
| 5           | LAST_SYNC_DURATION   | NUMBER             | Yes        | (null)         |
| 6           | LAST_SYNC_ROWS       | NUMBER             | Yes        | (null)         |
| 7           | LAST_SYNC_RPM        | NUMBER             | Yes        | (null)         |
| 8           | LAST_SYNC_BATCH_SIZE | NUMBER             | Yes        | (null)         |
| 9           | LAST_SYNC_BULK_LIMIT | NUMBER             | Yes        | (null)         |
| 10          | LAST_SYNC_SECS       | NUMBER             | Yes        | (null)         |
| 11          | LAST_SYNC_MAX_MINS   | NUMBER             | Yes        | (null)         |
| 12          | LAST_SYNC_RESULT     | VARCHAR2(10 BYTE)  | Yes        | (null)         |
| 13          | LAST_SYNC_ERROR      | VARCHAR2(500 BYTE) | Yes        | (null)         |

**Primary Key:** PK\_SYNC\_STATE\_DOC (SRC\_TABLE\_NAME)

## SYNC\_TRG\_DOC trigger

This database trigger is used to populate the SYNC\_STAGE\_DOC staging table whenever data is modified in the IN\_DOC table. It is used by the synchronization framework to determine which rows have changed in the source table and requires propagation to the destination table. This DML trigger will fire after each row is inserted or updated or deleted in the source table.

The SYNC\_TRG\_DOC trigger has following dependencies.

| ❖ OWNER | ❖ TYPE   | ❖ NAME                         |
|---------|----------|--------------------------------|
| INUSER  | PACKAGE  | <a href="#">IN_DB_UPGRADE</a>  |
| INUSER  | SEQUENCE | <a href="#">SYNC_SEQ_DOC</a>   |
| INUSER  | TABLE    | <a href="#">IN_DOC</a>         |
| INUSER  | TABLE    | <a href="#">IN_INSTANCE</a>    |
| INUSER  | TABLE    | <a href="#">SYNC_STAGE_DOC</a> |
| PUBLIC  | SYNONYM  | <a href="#">ALL_OBJECTS</a>    |

## SYNC\_SEQ\_DOC sequence

The SYNC\_TRG\_DOC trigger uses this sequence to generate the primary key (STAGE\_ID) when populating the SYNC\_STAGE\_DOC table.



Testing was conducted with row counts between 0 and 90 million rows in the SYNC\_STAGE\_DOC table and up to 465 million rows across the source and destination tables. Testing was conducted with LOGGING enabled for all objects.

According to an Oracle Active Session History (ASH) Report the following indexes had the largest impact towards degrading the performance of data synchronization. The impact of each index can vary depending on the implementation and usage of the available document keys.

NEW\_DOC\_IDX2 - 19.57 % - db file sequential read (DRAWER\_ID, FOLDER, TAB, F3, F4, F5, DOC\_TYPE\_ID)

NEW\_DOC\_IDX83 - 16.09 % - db file sequential read (F3)

NEW\_DOC\_IDX84 - 10.18 % - db file sequential read (F4)

NEW\_DOC\_IDX82 - 3.49 % - db file sequential read (TAB)

NEW\_DOC\_IDX85 - 2.81 % - db file sequential read (F5)

**The following synchronization throughput was recorded during testing**

- With no indexes on the destination table other than the PK and PKCI indexes, which should be in place for efficient row lookup and synchronization.

**Throughput was between 800K - 1.2M Rows Per Minute (RPM)**

- With all indexes except NEW\_DOC\_IDX2, NEW\_DOC\_IDX83, NEW\_DOC\_IDX84.

**Throughput was between 70K - 212K Rows Per Minute (RPM)**

- With all indexes having been created on the destination table.

**Throughput was between 3K - 7K Rows Per Minute (RPM)**

Due to the significant increase in the duration of the synchronization process when indexes are in place, it is advised to delay index creation until closer to the scheduled downtime event. You may also conduct your own internal benchmarking to measure the effects and choose to subsequently remove the indexes or the entire synchronization framework until the days or hours leading up to the scheduled downtime.