Oracle® Argus Mart

Extensibility Guide

Release 8.2

E96565-01

August 2018

This document contains information that helps a database expert to extend Argus Mart data structure or logic for custom data.

1 Assumptions

This guide assumes the following:

- Argus Mart (AM) is dependent on Argus Safety (AS).
- You should avoid executing any such functionality that may be implemented in a simpler and more general way in Argus Safety.
- It is not necessary to wrap PL/SQL components of the ETL.

2 Introduction to Argus Mart ETL

Argus Mart is a data source for analysis and reporting in medical product safety and pharmacovigilance. The primary data is the adverse event cases managed by the Argus Safety application. Argus Mart consists of one or more data marts and the software to `Extract` the source data from Argus Safety; `Transform` the data, and `Load` it into the Argus Mart data marts. The two sets of tables used in Argus Mart are:

- Reporting Mart (RM)
- Signal Mart (SM)

The data can be customized before or after fetching data from Argus Safety database with the help of ETL Hooks. The next section comprises the details about ETL Hooks.

3 ETL Hooks

ETL Hooks are the custom procedures that are used to extend the existing functionality or behavior of the Argus Mart ETL.

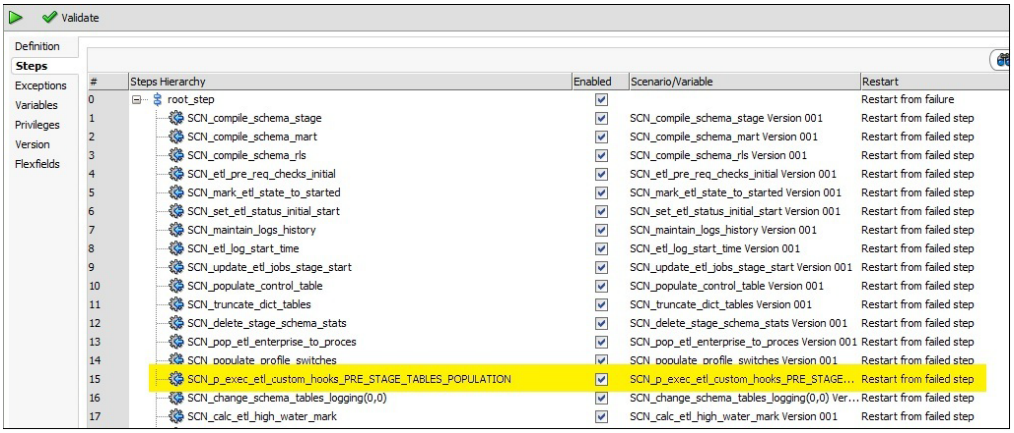
ETL Hooks can be executed to perform actions while fetching data from the source database to the destination database, as required by the users.

The data fetched from Argus Safety database can be modified before or after uploading the data into the Argus Mart database. You can populate customized (non-standard/user-defined) columns to the standard RM and SM tables. Besides, you can also populate customized table(s).

For more information on populating customized columns or tables, refer to Section 5, "Customize Tables".

ETL Hooks perform these actions at different stages of both initial and incremental ETL.

The ODI Load Plan lists all the ETL scenarios, as shown below:



3.1 List of Argus Mart ETL Hooks

Argus Mart comprises the following ETL Hooks that can be executed at different stages:

- CUSTOM ROUTINE BEFORE STAGE TABLES POPULATION—Execute before populating the Staging tables.
- CUSTOM ROUTINE BEFORE REPORTING TABLES POPULATION—Execute before populating the Reporting tables (or RM tables).
- CUSTOM ROUTINE AFTER REPORTING TABLES POPULATION—Execute after populating the Reporting tables (or RM tables).
- CUSTOM ROUTINE BEFORE SIGNAL HELPER TABLES POPULATION—Execute before populating the Signal Helper tables.
- CUSTOM ROUTINE AFTER SIGNAL HELPER TABLES POPULATION—Execute after populating the Signal Helper tables.
- CUSTOM ROUTINE AFTER ETL—Execute after running the initial or incremental ETL (post-ETL commit).

> **Note:** ETL hooks CUSTOM ROUTINE BEFORE SIGNAL HELPER TABLES POPULATION and CUSTOM ROUTINE AFTER SIGNAL HELPER TABLES POPULATION are executed only when any case revision is being processed for SM tables in the current ETL run.

3.2 View Argus Safety User Interface

ETL Hooks help in customizing the existing ETL to suit your requirements through Argus Safety user interface.

To view the Argus Safety user interface:

1. Log in to the Argus Safety application.
2. From the menu bar, click **Argus Console**.

The menu bar updates for Argus Console.

3. From the new menu bar, click **System Configuration**.
4. Click **System Management (Common Profile Switches)**.

   On the left hand side of the screen, a list of all system-managed common profile switches appears.
5. Expand the Common Profile folder and click **Argus Mart**.

   The **Modify Argus Mart** page appears.

> **Note:** ETL hooks are the global-level switches, visible on Argus Safety Console when you are logged-in through the default enterprise. These switches are visible only after installing and creating Argus Mart schema.

3.3 Execute ETL Hooks

The ETL Hooks can be executed at different staging levels.

1. Before populating the reporting tables, execute CUSTOM ROUTINE BEFORE STAGE TABLES POPULATION, and/or CUSTOM ROUTINE BEFORE REPORTING TABLES POPULATION.
2. Before populating the signal tables, execute ETL Hooks CUSTOM ROUTINE AFTER REPORTING TABLES POPULATION, CUSTOM ROUTINE BEFORE SIGNAL HELPER TABLES POPULATION, and/or CUSTOM ROUTINE AFTER SIGNAL HELPER TABLES POPULATION.
3. After populating reporting and signal tables and before populating the results, execute CUSTOM ROUTINE AFTER ETL.

Figure 1 Sequence of ETL Hook Execution

To execute an ETL Hook:

1. On the **Modify Argus Mart** page, enter a procedure name in the text box relevant to the ETL Hook execution stage. The ETL type can be initial or incremental where

the customized data will be fetched. The ETL Hook will look for the database object that matches the procedure name.

A procedure name denotes the ETL Hook that comprises the custom procedure, or steps. Based on the stage (initial or incremental) where this procedure is being called, the data is modified before sending it to the destination tables.

2. If the database object (or procedure) is found, the ETL is executed as follows:
   - If the ETL executes without any errors, then move to the next procedure.
   - If the ETL executes with errors, then log the error(s) and exit.

     When executing ETL, if there is any error while populating the staging schema tables, error(s) are logged in the table ETL_STAGE_LOG, whereas while populating the mart schema tables, error(s) are logged in the table ETL_MART_LOG.
3. If the database object (or procedure) is not found, then log the error(s) and fail the ETL. In this case, the ETL may be executed if you have explicitly created an exception-handling for such cases to absorb any exceptions and move on to the next procedure.

   Optionally, to resolve this issue, create a procedure of that name, provide an existing procedure name, or remove the configuration.

WARNING:
- **You should not modify the existing data or names of the objects. Although additional objects can be created, all changes should first be tested on a test environment, before implementing them in a production environment.**
- **No data commit should be done within ETL Hooks.**

4 Configure Flexible Data Re-categorization Code List

Flexible Data Re-categorization is an Argus Safety functionality through which users can define code list display values (LM tables) in different languages.

Argus Safety maintains this data in the tables CODE_LIST_MASTER, CODE_LIST_CODE_ATTRIBUTES, and CODE_LIST_DETAIL_DISCRETE for supported languages.

The Argus Mart ETL populates the code list data in the tables RM_CODE_LIST_MASTER, RM_CODE_LIST_CODE_ATTRIBUTES, and RM_CODE_LIST_DETAIL_DISCRETE, which are used for processing the SM table columns.

Based on the applicable language, the SM table fields are divided into the following categories:

- **LI - Language Independent fields**: These fields are applicable for first-human language, or second-human language parameters. The values are the same, irrespective of the language code.
- **FL - First Language fields**: These fields are populated on the basis of the Argus Mart Common Profile Switch FIRST_HUMAN_LANGUAGE, which is present in Argus Console.

  The default value of this switch is **en**. As a result, the default language in which these fields will be populated is English.

If the user configures this switch with any other available language and if its corresponding data is also present in the Flexible Data Re-categorization code list tables, these first language fields will contain data as per the configured language.

- **SL - Second Language fields**: These fields are populated on the basis of the Argus Mart Common Profile Switch SECOND_HUMAN_LANGUAGE, which is present in Argus Console.

  The default value of this switch is NULL. This implies that the second language fields will be populated in the SM tables only after this switch is configured.

  If the user configures this switch with any other configured language (like German), and:

  - If its corresponding data is also present in the Flexible Data Re-categorization code list tables, these second language fields will contain data as per the configured language.
  - If its corresponding data is not available in the Flexible Data Re-categorization code list tables, these second language fields will contain data as per the English language.
- **EN - English fields**: These fields are always populated for only the English language.
- **J - Japanese fields**: These fields are applicable only if the second-human language parameter is set to Japanese language.
- **EN_ABBRV**: These fields use the EN_ABBRV language for the specified code list.
- **SM:** These fields use the SM language for the specified code list.

The following is an example to configure a code list display value in a new language for an already existing code in Argus Safety:

Example 1 Configure Flexible Data Re-categorization

For a code list GENDER, data in the table CODE_LIST_DETAIL_DISCRETE for code 1 is available in the following four decode contexts (languages):

Figure 2 Original Decode Contexts (Languages)

CODE_LIST_ID	DECODE_CONTEXT	CODE	DISPLAY_VALUE	PREFERRED	SORT	LAST_UPDATE_TIME	ENTERPRISE_ID
GENDER	en	1	Male	0	(null)	05-FEB-13	1
GENDER	ja	1	男性	0	(null)	05-FEB-13	1
GENDER	E2B	1	1	0	(null)	05-FEB-13	1
GENDER	SM	1	M	0	(null)	05-FEB-13	1

To configure the same code 1 in the code list GENDER for a new language such as GERMAN (decode context **ge**), populate the table CODE_LIST_DETAIL_DISCRETE with required values in the GERMAN language.

```
INSERT INTO CODE_LIST_DETAIL_DISCRETE (CODE_LIST_ID, DECODE_CONTEXT, CODE,
DISPLAY_VALUE, PREFERRED, SORT, LAST_UPDATE_TIME, ENTERPRISE_ID) VALUES (
'GENDER', 'ge', 1, 'männlich', 0, null, sysdate, 1);
```

Figure 3 New Decode Contexts (Languages)

CODE_LIST_ID	DECODE_CONTEXT	CODE	DISPLAY_VALUE	PREFERRED	SORT	LAST_UPDATE_TIME	ENTERPRISE_ID
GENDER	en	1	Male	0	(null)	05-FEB-13	1
GENDER	ja	1	男性	0	(null)	05-FEB-13	1
GENDER	E2B	1	1	0	(null)	05-FEB-13	1
GENDER	SM	1	M	0	(null)	05-FEB-13	1
GENDER	ge	1	männlich	0	(null)	20-FEB-13	1

5 Customize Tables

You can customize RM and SM tables.

5.1 Customize RM Tables

5.1.1 Populate a Column Added in the Argus Safety table

You can populate new column(s) to existing RM tables that are added to the Argus Safety tables.

To populate the new column:

1. Check mapping between the source and the target tables, through the table *ETL_SIGNAL_TABLE_MAPPING*.
2. Based on the mapping between the Argus Safety and RM tables, fetch the desired column in the RM table through ETL Hooks.

There can be different source types to fetch data into tables at initial or incremental stages, such as:

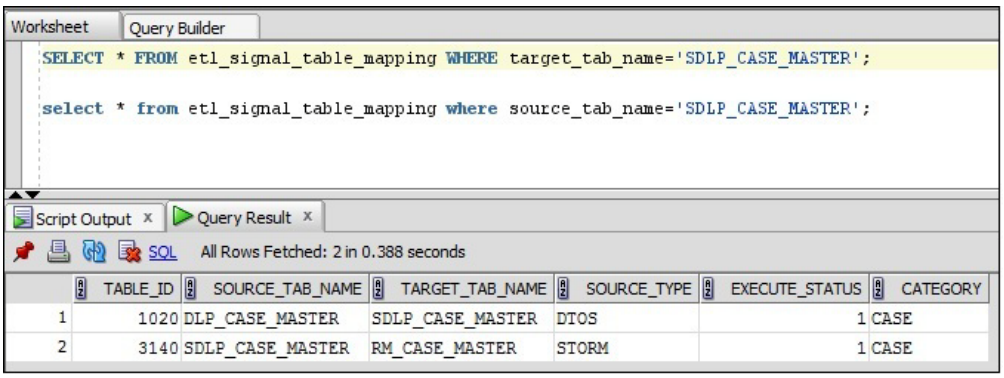
- ATOS - Argus Safety to Staging tables
- DTOS - DLP to Staging tables
- STORM - Staging to RM tables
- STOSM - PRE_SM to SM Tables

Example 2 Populate a new column in the RM table

Follow the steps given below to populate a new column in the table RM_CASE_MASTER:

1. Check the mapping between RM_CASE_MASTER, and the source table (SDLP_CASE_MASTER) to fetch columns from the staging table into the RM table. In this case, the source type is STORM.
2. Check the source table for target table (SDLP_CASE_MASTER), where columns are fetched from Argus Safety into the staging table. In this case, the source type can be ATOS/DTOS.

Figure 4 Populating New Columns in the RM Table



3. Once the mapping of the tables has been identified, fetch these column(s) from Argus Safety/DLP into Argus Mart, as required.

5.1.2 Populate a Calculated Column in RM

You can populate customized calculated column(s) to the existing RM tables that do not exist in the Argus Safety database. You can also populate column(s) which are calculated on the basis of the existing columns in a table.

To populate such custom columns, create a procedure, and use ETL Hooks to execute them.

Example 3 Create a custom procedure

Creating a procedure called P_UPD_RM_CASE_PAT_INFO, to populate a new column called PAT_FULL_NAME in the table RM_CASE_PAT_INFO. The value of this column is calculated, and populated from the existing columns of RM_CASE_PAT_INFO. Here, we concatenate three columns PAT_INITIALS, PAT_FIRSTNAME, and PAT_LASTNAME to populate this value as PAT_FULL_NAME.

1. Create a procedure called P_UPD_RM_CASE_PAT_INFO.
2. In the table RM_CASE_PAT_INFO, populate a new column called PAT_FULL_NAME.

   The value of this column is calculated, and populated from the existing columns of RM_CASE_PAT_INFO.
3. Concatenate three columns PAT_INITIALS, PAT_FIRSTNAME, and PAT_LASTNAME to populate this value as PAT_FULL_NAME.

Figure 5 Create Custom Procedure

```
CREATE OR REPLACE PROCEDURE p_upd_rm_case_pat_info AS
BEGIN
  UPDATE rm_case_pat_info
     SET pat_full_name = pat_initials || ' ' || pat_firstname || ' ' || pat_lastname
   WHERE pat_full_name IS NULL;
EXCEPTION
  WHEN OTHERS THEN
    pkg_sm_logging.p_etl_mart_log (SUBSTR (SQLERRM, 1, 300),
                                   'p_upd_rm_case_pat_info',
                                   'Error during updation of RM_CASE_PAT_INFO.');
END p_upd_rm_case_pat_info;
/
```

4. Once this procedure is created, call this procedure using ETL Hook PRE_REPORTING_TABLES_POPULATION.

   The ETL Hook will populate this new column PAT_FULL_NAME in SDLP_CASE_PAT_INFO.

5.2 Customize SM Tables

When using ETL Hooks to populate data into SM tables, it is recommended to use RM tables as the source, with join(s) to the chunk table ETL_SM_CASES_TO_PROCESS_CHUNK.

Do not use RM Views such as V_ETL_CASE_MASTER, because each view is joined individually to the chunk table. This can lead to sluggish performance of the application if ETL Hook extracts data from the multiple views.

Example 4 Populate data into SM tables using the chunk table

In the example given below, two RM tables RM_CASE_MASTER and RM_CASE_PAT_INFO have been joined with the chunk table ETL_SM_CASES_TO_PROCESS_CHUNK to fetch data into the SM table with the help of the ETL Hooks.

```
SELECT *
  FROM rm_case_master rcm, rm_case_pat_info rcpi, etl_sm_cases_to_process_chunk chnk
 WHERE chnk.enterprise_id = rcm.enterprise_id
   AND chnk.case_id = rcm.case_id
   AND chnk.validstart >= rcm.effective_start_date
   AND chnk.validstart < rcm.effective_end_date
   AND chnk.enterprise_id = rcpi.enterprise_id
   AND chnk.case_id = rcpi.case_id
   AND chnk.validstart >= rcpi.effective_start_date
   AND chnk.validstart < rcpi.effective_end_date;
```

5.2.1 Define a New Column in SM table

New column(s) can be added before or after fetching data into SM tables. The columns to be added into SM can have simple or complex calculations.

- Simple calculation-based columns: SM Views are created based on Argus Safety tables. When a customized column having simple calculations is to be added in an SM table, these calculations are done on SM Views, such as:
  - select <column-name> from <table-name> table where <column-name> DateOfBirth = '1990';
  - The definition of these SM Views is based on RM tables and is not updated automatically. When calculations are done, make sure to update the views, to reflect the changes. Once SM Views are updated, use ETL Hooks to fetch this column(s) into SM table, as required.
- Complex calculation-based columns: *PRE_SM* table is a set of tables that comprises the staging data for SM tables. You can perform complex calculations on these tables and then fetch them into SM tables using ETL Hooks.

The basic procedure to define a new column is the same for both simple and complex calculation-based column(s) - SM View and PRE_SM tables.

Simple and complex calculations are segregated to improve performance of execution of the data.

5.2.2 Populate a Custom UVT table

The Argus Mart ETL maintains Unique Value Tables (UVT) for first-human language and second-human language, as defined in the system. The UVTs contain the list of distinct values available for categorical data items in the case data.

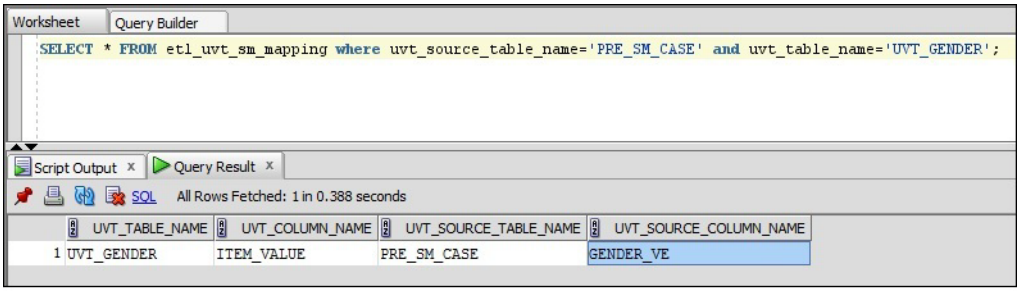
To populate a custom UVT table, you must first check the source and target table mapping from the table *ETL_UVT_SM_MAPPING*.

Fetch the data from the source table into the target table as per the mapping done above.

Example 5 Populate a custom UVT table

Follow the steps given below to populate a column in the table UVT_GENDER:

1. Check the mapping between the source table PRE_SM_CASE and the target table UVT_GENDER from the table ETL_UVT_SM_MAPPING.



2. After mapping the tables and their related columns, you can populate the column(s), as required.

6 Customize Table for CLOB columns

To reduce the time taken to load the CLOB data from Argus Safety source tables to Argus Mart stage tables, add or modify the data in the table ETL_SPLIT_CLOB_COLUMN_MAP available in Argus Mart schema.

With this customization, the ETL fetches only the first 32767 bytes of the source CLOB column to the respective column of the stage table. The value in the source CLOB column splits into twenty five VARCHAR2 columns at the source before moving the data to the respective stage column. Hence, reduced loading time of these tables, and reduced ETL time.

Note:

- Enabling this feature fetches only the first 32767 bytes of the data from the source CLOB column to the respective stage table column. The rest of the data is not copied.
- Enable this feature purely based on your own discretion.

The following are the table details to enable or disable this customization:

COLUMN NAME	DATA TYPE (LENGTH)	DESCRIPTION
TABLE_NAME	VARCHAR2 (30)	Defines Argus Safety source table name from which data is populated in to Argus Mart stage table. For example, DLP_CASE_NARRATIVE.
COLUMN_NAME	VARCHAR2 (30)	Defines the name of the CLOB column present in the source table. For example, ABBREV_NARRATIVE.
ENABLE_YN	VARCHAR2 (1)	Defines the values for this column. Y—The feature is enabled N—The feature is disabled

Besides, as part of the Factory Data, the following columns of the table ETL_SPLIT_CLOB_COLUMN_MAP are disabled by default (ENABLE_YN = N):

TABLE_NAME	COLUMN_NAME	ENABLE_YN
CMN_REG_REPORTS	NULLIFICATION_REASON	N
DLP_CASE_ASSESS	BFARM_MANUAL_TEXT	N
	EVALUATION	N
	EVALUATION_J	N
DLP_CASE_DOSE_REGIMENS	DOSE_DESCRIPTION	N
	DOSE_DESCRIPTION_J	N
DLP_CASE_EU_DEVICE	CORRECTIVE_ACTION	N
	CORRECTIVE_ACTION_FINAL	N
	COUNTRIES_OF_DISTRIBUTION	N
	CURRENT_DEV_LOCATIONS	N
	FURTHER_INVESTIGATION	N
	INDENTIF	N
	INVESTIGATION_RESULT	N
	MANUFACTURER_COMMENTS	N
	PROJECTED_TIMING	N
	PROJECTED_TIMING_FINAL	N
DLP_CASE_EVENT	DETAILS	N
	DETAILS_J	N
DLP_CASE_FOLLOWUP	JUSTIFICATION	N
	JUSTIFICATION_J	N
DLP_CASE_JUSTIFICATIONS	J_TEXT	N
	J_TEXT_J	N

TABLE_NAME	COLUMN_NAME	ENABLE_YN
DLP_CASE_LAB_DATA	COMMENTS	N
	COMMENTS_J	N
	NOTES	N
	NOTES_J	N
DLP_CASE_NARRATIVE	ABBREV_NARRATIVE	N
	ABBREV_NARRATIVE_J	N
DLP_CASE_NOTES_ATTACH	NOTES	N
	NOTES_J	N
DLP_CASE_PAT_HIST	NOTE	N
	NOTE_J	N
DLP_CASE_PAT_INFO	NOTES	N
	NOTES_J	N
DLP_CASE_PROD_DEVICES	NARRATIVE_TEXT	N
	PRELIMINARY_COMMENTS	N
	PRELIMINARY_COMMENTS_J	N
DLP_CASE_PRODUCT	COMMENTS_TIMEFRAME	N
	GENERIC_NAME	N
	GENERIC_NAME_J	N
	NOTES	N
	NOTES_J	N
	QC_ANALYSIS_CAT_TEXT	N
	QC_ANALYSIS_CAT_TEXT_J	N
	QC_ANAL_SUMMARY_TEXT	N
	QC_ANAL_SUMMARY_TEXT_J	N
	QC_COMMENT	N
	QC_COMMENT_J	N
	QC_COMPLAINT_CAT_TEXT	N
	QC_COMPLAINT_CAT_TEXT_J	N
	QC_RESULT	N
	QC_RESULT_J	N
	REASON_DOWNGRADE	N
	RETRO_INFECTION	N
DLP_CASE_REPORTERS	NOTES	N
	NOTES_J	N
DLP_CASE_ROUTING	COMMENT_TXT	N
	COMMENT_TXT_J	N

TABLE_NAME	COLUMN_NAME	ENABLE_YN
DLP_CASE_STUDY	STUDY_DESC	N
	STUDY_DESC_J	N
LM_LABELED_TERMS	NOTES	N
	NOTES_J	N
LM_PRODUCT	PROD_GENERIC_NAME	N
	PROD_GENERIC_NAME_J	N
RPT_ROUTING	COMMENT_TXT	N

To enable this feature:

- For the columns available in the table ETL_SPLIT_CLOB_COLUMN_MAP as part of the factory data, update the column value for ENABLYE_YN as **Y** for the required column. For example:

```
UPDATE ETL_SPLIT_CLOB_COLUMN_MAP
   SET ENABLYE_YN = 'Y'
WHERE TABLE_NAME = '<TABLE_NAME>'
   AND COLUMN_NAME = '<COLUMN_NAME>'
```

- For new columns that are not available in the table ETL_SPLIT_CLOB_COLUMN_MAP as part of the factory data, create new entry in this table with the column value for ENABLYE_YN as **Y** for the required column. For example:

```
INSERT INTO ETL_SPLIT_CLOB_COLUMN_MAP
       (TABLE_NAME , COLUMN_NAME , ENABLYE_YN)
VALUES ('<TABLE_NAME>, '<COLUMN_NAME>', 'Y');
COMMIT;
```

> **Note:** Enabling this feature only affects the data that is fetched from Argus Safety Source to Argus Mart in the Next ETL. It does not affects the existing data in the MART. (It does not triggers the re-load of data for the respective CLOB columns.)

To disable this feature:

Set the column value for ENABLYE_YN as **N** for the required column. For example:

```
UPDATE ETL_SPLIT_CLOB_COLUMN_MAP
   SET ENABLYE_YN = 'N'
WHERE TABLE_NAME = '<TABLE_NAME>'
   AND COLUMN_NAME = '<COLUMN_NAME>'
```

> **Note:** Disabling this feature only affects the data that is fetched from Argus Safety Source to Argus Mart in the Next ETL. It does not affects the existing data in the MART. (It does not triggers the re-load of data for the respective CLOB columns.)

7 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.

Oracle Argus Mart Security Configuration Guide, Release 8.2
E96565-01