Oracle® Retail AI Foundation Cloud Services

Operations Guide

ORACLE®

Oracle Retail AI Foundation Cloud Services Operations Guide, Release 22.2.302.0

F61893-01

Contents

3   AI Foundation Cloud Services Standalone Processes

4   Retail Insights Standalone Process Flows

5   Data Validation Framework

6   Support Utilities

List of Tables

Send Us Your Comments

Oracle Retail Insights and AI Foundation Cloud Services Operations Guide

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:**
>
> Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at `http://www.oracle.com`.

Preface

This Operations Guide provides critical information about the processing and operating details of the Retail Insights and AI Foundation Cloud Services, including the following:

- Standalone and Adhoc batch processes
- Integration processes

Audience

This guide is for:

- Systems administration and operations personnel
- Systems analyst
- Integrators and implementers

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.

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Oracle Help Center (docs.oracle.com)

Oracle Retail Product documentation is available on the following website https://docs.oracle.com/en/industries/retail/html

Comments and Suggestions

Please give us feedback about Oracle Retail Help and Guides. You can send an e-mail to:
retail-doc_us@oracle.com

Oracle Retail Cloud Services and Business Agility

The Oracle Retail Insights and AI Foundation Cloud Services are hosted in the Oracle Cloud with the security features inherent to Oracle technology and a robust data center classification, providing significant uptime. The Oracle Cloud team is responsible for installing, monitoring, patching, and upgrading retail software.

Included in the service is continuous technical support, access to software feature enhancements, hardware upgrades, and disaster recovery. The Cloud Service model helps to free customer IT resources from the need to perform these tasks, giving retailers greater business agility to respond to changing technologies and to perform more value-added tasks focused on business processes and innovation.

Oracle Retail Software Cloud Service is acquired exclusively through a subscription service (SaaS) model. This shifts funding from a capital investment in software to an operational expense. Subscription-based pricing for retail applications offers flexibility and cost effectiveness.

1
Introduction

This document is intended to guide a Retail Analytics and Planning Cloud Services implementer through the internal operations of key areas of the AI Foundation platform that they will need to interact with during a project, such as ad hoc batch processes and integration programs. All programs are located within the Process Orchestration and Monitoring (POM) application and the reader is expected to be familiar with that tool.

This guide includes the following topics:

- **Retail Insights Standalone Batch Processes** - This chapter provides an overview of each Retail Insights batch program or process flow in the Standalone set of jobs in POM, the input and output tables involved in the process, and any dependencies or usage details to consider before running them.

- **Retail AI Foundation Cloud Services Standalone Batch Processes** - This chapter provides an overview of each AI Foundation Cloud Services batch program or process flow available in the Standalone set of jobs in POM. The primary purpose of the AI Foundation ad hoc programs is to integrate data from either RI, flat files, or Innovation Workbench.

- **Retail Insights Standalone Process Flows** - This chapter provides a set of cross-reference tables showing how programs in the RI standalone processes are linked to each other, such as the staging and load jobs to move a single file into the database from start to finish. This should be used to disable all unneeded jobs in the adhoc load processes for files you are not trying to load.

- **Data Validation Framework** - This chapter explains the data validation procedures associated with foundation input files. The data validation framework checks for common mistakes and issues in the incoming data files and either fails the process or outputs warnings to the database, depending on the issue.

- **Support Utilities** - This chapter describes the self-service utilities used for environment maintenance and cleanup. Implementers should be aware of the utilities available to them and leverage them during the project, as needed.

2
Retail Insights Standalone Processes

The primary function of standalone processes in Retail Insights (RI) is to load history data in a new environment for use in one or more applications on the platform. These process flows group together multiple, related programs that load data files, stage them in the database, and transform them into multiple target tables in the RI data warehouse. Processes are also available for integrations with Merchandise Financial Planning (MFP) and Xstore.

Adjustments History Load

Module Name	HIST_CSV_ADJUSTMENTS_LOAD_ADHOC
Description	Loads the `ADJUSTMENT.csv` file into RI and populates key data tables used to integrate with other systems for history data.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

The history load process for Inventory Adjustment transactions accepts an input file at the transaction level using the file specification for `ADJUSTMENT.csv`. It assumes the file has already been moved into place using the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day), as well as the week aggregate used for integrations (item/location/week). The Reason dimension is also seeded with records if the reason code and reason description are provided on the transactions.

> ✎ **Note:**
>
> This process does not currently populate BI aggregate tables. Those jobs need to be run separately after each execution of this process if it is necessary to use this data for reporting in RI.

Key Tables Affected

Table	Usage
W_ADJUSTMENT_FTS	File Input
W_REASON_DS	Staging
W_RTL_INVADJ_IT_LC_DY_FS	Staging
W_RTL_INVADJ_IT_LC_DY_F	Output (Base Fact)
W_RTL_INVADJ_IT_LC_WK_A	Output (Aggregate)

Deal Income History Load

Module Name	HIST_CSV_DEAL_INCOME_LOAD_ADHOC
Description	Loads the `DEAL_INCOME.csv` file into RI and populates key data tables used to integrate with other systems for history data.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

The history load process for Deal Income transactions accepts an input file at the transaction level using the file specification for `DEAL_INCOME.csv`. It assumes the file has already been moved into place using the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, and then loads it into the base fact (item/location/day) as well as the week aggregate used for integrations (item/location/week).

Key Tables Affected

Table	Usage
W_RTL_DEALINC_IT_LC_DY_FTS	File Input
W_RTL_DEALINC_IT_LC_DY_FS	Staging
W_RTL_DEALINC_IT_LC_DY_F	Output (Base Fact)
W_RTL_DEALINC_IT_LC_WK_A	Output (Aggregate)

Default Calendar Initialization

Module Name	AUTO_GEN_CALENDAR_LOAD_ADHOC
Description	Automatically generates a generic NRF fiscal calendar and sets up the RI database with it.
Dependencies	None
Business Activity	Initial System Setup

Design Overview

The auto-generated calendar process does not require any input files. Instead, it uses an internal calendar definition based on the National Retail Federation (NRF) 4-5-4 business calendar to populate the Retail Insights data model with basic calendar information. The NRF calendar typically starts around the first week of February and runs for 52 or 53 weeks, depending on the year. The default calendar start froms January 2017 and extends for approximately 30 years. It automatically includes 53-week years where appropriate and follows the NRF guidelines for fiscal weeks and periods.

This process performs all the necessary transform and load jobs required to set up the RI calendar. This process should only be used if you cannot get a business calendar definition from any other source, and the retailer does not want to provide a file themselves. Once this process runs, you can disable `W_MCAL_PERIOD_DS_JOB` in your nightly batch if you do not intend to ever provide a calendar file directly.

This process also populates the Gregorian system calendar at the same time the fiscal calendar is loaded. The Gregorian calendar requires additional start and end date parameters from `C_ODI_PARAM` to define the time range to generate. It must be greater than the range of time in the fiscal calendar. Output tables that start with `W_MCAL_` are mainly used for fiscal calendar generation, while the other tables, such as `W_DAY_D`, are used for the Gregorian calendar. All output tables must be successfully populated with calendar data to use the platform.

Key Tables Affected

Table	Usage
W_MCAL_PERIOD_DS	Staging
W_TIME_OF_DAY_D	Output
W_DAY_D	Output
W_YEAR_D	Output
W_QTR_D	Output
W_MONTH_D	Output
W_WEEK_D	Output
W_MINUTE_OF_DAY_D	Output
W_MCAL_CONFIG_G	Output
W_MCAL_CAL_D	Output
W_MCAL_PERIOD_D	Output
W_MCAL_DAY_D	Output
W_MCAL_WEEK_D	Output
W_MCAL_YEAR_D	Output
W_MCAL_QTR_D	Output
W_RTL_MCAL_DAY_SHIFT_D	Output
W_RTL_MCAL_DAY_UNSHIFT_D	Output
W_RTL_MCAL_DAY_GUNSHIFT_D	Output
W_RTL_MCAL_DAY_CUSTOM_D	Output
W_RTL_MCAL_WEEK_SHIFT_D	Output
W_RTL_MCAL_WEEK_UNSHIFT_D	Output
W_RTL_MCAL_PERIOD_SHIFT_D	Output
W_RTL_MCAL_PERIOD_UNSHIFT_D	Output

ETL Business Date Update

Module Name	LOAD_CURRENT_BUSINESS_DATE_ADHOC

Description	Override the current business date used for loading data into RI.
Dependencies	None
Business Activity	Batch Administration

Design Overview

This process updates the business date in the Retail Insights data model to prepare the batch infrastructure for loading additional data on this date. This process should be used during the history and seeding data loads to align the current RI system date with the date on the input data files. The system date must match with the incoming data for positional files such as inventory and pricing when you are doing seed loads to initialize the system. For transactional data loads, it is only necessary to have the system date be on or after the latest date in the file, because RI supports back-posting transaction records to prior dates.

Key Tables Affected

Table	Usage
W_RTL_CURR_MCAL_G	Output

History Data Cleanup

Module Name	HIST_DATA_CLEANUP_ADHOC
Description	Erase all data from Inventory and Price tables in RI, in order to restart your history load for those interfaces.
Dependencies	None
Business Activity	Historical Data Load

Design Overview

This process erases all data from select functional areas (currently Inventory Position and Pricing facts). The purpose of the process is to reset the environment if the data currently loaded is invalid or unwanted, and you'd like to start over with empty tables.

Note that

> **Note:**
>
> It does not erase partition structures, so you will need to load data for the same range of dates already available.
>
> It also does not reset the `C_HIST_LOAD_STATUS` table, so you will need to update that before loading any new data.

History Data File Upload

Module Name	HIST_ZIP_FILE_LOAD_ADHOC
Description	Looks for the `RIHIST_RMS_DATA.zip` file and unpacks it, moving any files to the incoming directory for batch processes.
Dependencies	None
Business Activity	Historical Data Load

Design Overview

This process moves and unloads a ZIP file (specifically `RIHIST_RMS_DATA.zip`) so that the file contents may be used for one or more history and seeding load jobs. The ZIP file may contain one or multiple files. This process is a prerequisite to running any history or seeding load programs.

The first job in this process waits a set period of time for the ZIP file to be uploaded, and it fails if it is not received in that time (4 hours by default). The second job moves the ZIP file to the internal server location and unzip it. It deletes any files previously in the destination folder, unzip the new file, and move the ZIP file to an archive when complete. It fails if the ZIP does not contain any data files, as there is nothing for it to move.

Initial Base Cost Seeding

Module Name	SEED_CSV_W_RTL_BCOST_IT_LC_DY_F_PROCESS_ADHOC
Description	Loads a full snapshot of base cost data from `COST.csv` to initialize the positional data before a nightly batch can be enabled.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Nightly Batch Preparation

Design Overview

The seeding load process for Base Cost data accepts an input file at the item-location-date-supplier level using the file specification for `COST.csv`. It assumes the file has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day). This process is only for the base cost, a separate process loads the net cost, if required.

> **Note:**
>
> Seeding processes require a full snapshot of data for a single date, which covers all item/location combinations that should have a starting position for this fact. The seeding process must load data for the day before the nightly batch is going to run. Alternatively, you can include the full snapshots of data in your very first nightly batch and skip the seeding steps. This causes the nightly batch to take a significantly longer time to execute but avoids the manual load processes for all the positional facts.

Key Tables Affected

Table	Usage
W_COST_FTS	File Input
W_RTL_BCOST_IT_LC_DY_FS	Staging
W_RTL_BCOST_IT_LC_G	Output
W_RTL_BCOST_IT_LC_DY_F	Output

Initial Base Cost Seeding (Legacy)

Module Name	SEED_W_RTL_BCOST_IT_LC_DY_F_PROCESS_ADHOC
Description	Loads a full snapshot of base cost data from `W_RTL_BCOST_IT_LC_DY_FS.dat` to initialize the positional data before a nightly batch can be enabled.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Nightly Batch Preparation

Design Overview

The seeding load process for Base Cost data accepts an input file at the item-location-date-supplier level using the file specification for `W_RTL_BCOST_IT_LC_DY_FS.dat`. It assumes the file has already been moved into place using the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day).

> **Note:**
>
> Seeding processes require a full snapshot of data for a single date, which covers all item/location combinations that should have a starting position for this fact. The seeding process must load data for the day before the nightly batch runs. Alternatively, you can include the full snapshots of data in your very first nightly batch and skip the seeding steps. This causes the nightly batch to take a significantly longer time to execute but avoids the manual load processes for all the positional facts.

Key Tables Affected

Table	Usage
W_RTL_BCOST_IT_LC_DY_FS	Staging
W_RTL_BCOST_IT_LC_G	Output
W_RTL_BCOST_IT_LC_DY_F	Output

Initial Calendar Load

Module Name	CALENDAR_LOAD_ADHOC
Description	Runs all calendar creation and load processes to set up or update the system and fiscal calendars in RI.
Dependencies	CALENDAR_STG_CSV_LOAD_ADHOC
Business Activity	Initial System Setup

Design Overview

The calendar load ad hoc process performs all the necessary transform and load jobs to set up the RI calendars. It takes as input:

1. A calendar data file loaded using either the CSV or DAT file format and associated load program

2. Optional last-year mapping files to define shifted and unshifted calendars when reporting on LY data

3. System calendar start and end dates in `C_ODI_PARAM`

The calendar data must be in the form of a fiscal calendar (for example, a 4-5-4 or 13-period calendar). It must be at the period level of detail (not the day level) and should include start and end date ranges for the period, quarter, and year levels on each record. RI currently supports a single, hard-coded calendar ID (Retail Calendar~41) that should be used in the file's first column (`MCAL_CAL_ID`). Optional mapping files for this-year-to-last-year mappings may be provided if the business uses a custom definition of LY in reporting and analytics. These mappings control which range of dates are returned when pulling LY metrics in RI, such as when a fiscal week in the current year should be mapped to a different week in LY. Default mappings are created by the process if no data is provided.

This process populates the Gregorian system calendar at the same time the fiscal calendar is loaded. The Gregorian calendar requires additional start and end date parameters from `C_ODI_PARAM` to define the time range to generate. It must be greater than the range of time in the fiscal calendar. The calendar generation process does not support a 53-week year as the starting year, so it's recommended to make the start date of the Gregorian calendar at least 1 year earlier than the start of the fiscal calendar, which avoids improperly formed data in the fiscal calendar if the 53-week year is the first year.

Output tables that start with `W_MCAL_` are mainly used for fiscal calendar generation, while the other tables such as `W_DAY_D` are used for the Gregorian calendar. All output tables must be successfully populated with calendar data in order to use the platform. Validate the data closely after running this process to ensure nothing is missing or incorrect in the generated calendar data.

Key Tables Affected

Table	Usage
W_MCAL_PERIOD_DS	Input
W_RTL_MCAL_DAY_SHIFT_DS	Input
W_RTL_MCAL_DAY_UNSHIFT_DS	Input
W_RTL_MCAL_DAY_GUNSHIFT_DS	Input
W_RTL_MCAL_WEEK_SHIFT_DS	Input
W_RTL_MCAL_WEEK_UNSHIFT_DS	Input
W_TIME_OF_DAY_D	Output
W_DAY_D	Output
W_YEAR_D	Output
W_QTR_D	Output
W_MONTH_D	Output
W_WEEK_D	Output
W_MINUTE_OF_DAY_D	Output
W_MCAL_CONFIG_G	Output
W_MCAL_CAL_D	Output
W_MCAL_PERIOD_D	Output
W_MCAL_DAY_D	Output
W_MCAL_WEEK_D	Output
W_MCAL_YEAR_D	Output
W_MCAL_QTR_D	Output
W_RTL_MCAL_DAY_SHIFT_D	Output
W_RTL_MCAL_DAY_UNSHIFT_D	Output
W_RTL_MCAL_DAY_GUNSHIFT_D	Output
W_RTL_MCAL_DAY_CUSTOM_D	Output
W_RTL_MCAL_WEEK_SHIFT_D	Output
W_RTL_MCAL_WEEK_UNSHIFT_D	Output
W_RTL_MCAL_PERIOD_SHIFT_D	Output
W_RTL_MCAL_PERIOD_UNSHIFT_D	Output

Initial Calendar Staging

Module Name	CALENDAR_STG_CSV_LOAD_ADHOC
Description	Stages the CALENDAR.csv file for the ad hoc calendar load programs.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Initial System Setup

Design Overview

This process looks for the `CALENDAR.csv` file placed on the server by a history ZIP file upload, moves it into a preprocessing table in RI, and transforms it for use in the `CALENDAR_LOAD_ADHOC` process.

Key Tables Affected

Table	Usage
W_MCAL_PERIOD_DTS	File Input
W_MCAL_PERIODS_DS	Output

Initial Calendar Staging (Legacy)

Module Name	CALENDAR_STG_LOAD_ADHOC
Description	Stages the `W_MCAL_PERIOD_DS.dat` file for the ad hoc calendar load programs.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Initial System Setup

Design Overview

This process looks for the `W_MCAL_PERIOD_DS.dat` file placed on the server by a history zip file upload and imports it to a staging table for use in the `CALENDAR_LOAD_ADHOC` process.

Key Tables Affected

Table	Usage
W_MCAL_PERIODS_DS	File Input

Initial Dimension Load

Module Name	LOAD_DIM_INITIAL_ADHOC
Description	Runs all core dimension load programs in RI to bring in any staged dimension data and transform it to RI's data model.
Dependencies	LOAD_EXT_DIM_INITIAL_SI_ADHOC
Business Activity	Historical Data Load

Design Overview

This process runs the dimension load programs needed to initialize the data model with the core dataset needed for history and seed loads. Not all dimensions supported by RI are part of the initial load process, only those that are used in some way for history or downstream

application processing. This process does not load the flat files or stage the data, it only runs the final transform and load steps in the database. The jobs used by the process are the same as the ones in the nightly batch.

The process may be run repeatedly to load different sets of dimensions across multiple file uploads. For example, you may upload `PRODUCT.csv` and `ORGANIZATION.csv`, run the file staging process, and run the initial dimension load to import those files. You may then provide a new ZIP upload containing `EXCH_RATE.csv` and run the stage and load processes again to load that data in. The table below describes all of the input and output tables used in this process.

Key Tables Affected

Input Table	Output Table
W_DOMAIN_MEMBER_DS_TL	W_DOMAIN_MEMBER_LKP_TL
W_EMPLOYEE_DS	W_EMPLOYEE_D
W_EXCH_RATE_GS	W_EXCH_RATE_G
W_INT_ORG_DS	W_INT_ORG_D
W_INT_ORG_DHS	W_INT_ORG_DH
W_PARTY_ATTR_DS	W_PARTY_ATTR_D
W_PARTY_ORG_DS	W_PARTY_ORG_D
W_PARTY_PER_DS	W_PARTY_PER_D
W_PROD_CAT_DHS	W_PROD_CAT_DH
W_PRODUCT_ATTR_DS	W_PRODUCT_ATTR_D
W_PRODUCT_DS	W_PRODUCT_D
W_REASON_DS	W_REASON_D
W_RTL_ALC_DETAILS_DS	W_RTL_ALC_DETAILS_D
W_RTL_BUYER_DS	W_RTL_BUYER_D
W_RTL_CHANNEL_DS	W_RTL_CHANNEL_D
W_RTL_CO_HEAD_DS	W_RTL_CO_HEAD_D
W_RTL_CO_LINE_DS	W_RTL_CO_LINE_D
W_RTL_CO_SHIP_METHOD_DS	W_RTL_CO_SHIP_METHOD_D
W_RTL_CO_SHIP_TYPE_DS	W_RTL_CO_SHIP_TYPE_D
W_RTL_COMP_STORE_DS	W_RTL_COMP_STORE_D
W_RTL_CONS_METADATA_GS	W_RTL_CONS_METADATA_G
W_RTL_COUPON_DS	W_RTL_COUPON_D
W_RTL_DIFF_GRP_DS	W_RTL_DIFF_GRP_D
W_RTL_DIFF_RNG_DS	W_RTL_DIFF_RNG_D
W_RTL_DISCOUNT_TYPE_DS	W_RTL_DISCOUNT_TYPE_D
W_RTL_IT_SUPPLIER_DS	W_RTL_IT_SUPPLIER_D
W_RTL_ITEM_GRP1_DS	W_RTL_ITEM_GRP1_D
W_RTL_LOC_STOCK_CNT_DS	W_RTL_LOC_STOCK_CNT_D
W_RTL_ORG_FIN_DS	W_RTL_ORG_FIN_D
W_RTL_PHASE_DS	W_RTL_PHASE_D

Input Table	Output Table
W_RTL_PO_DETAILS_DS	W_RTL_PO_DETAILS_D
W_RTL_PRICE_CLR_IT_LC_DS	W_RTL_PRICE_CLR_IT_LC_D
W_RTL_PRODUCT_ATTR_DS	W_RTL_PRODUCT_ATTR_D
W_RTL_PRODUCT_BRAND_DS	W_RTL_PRODUCT_BRAND_D
W_RTL_PROMO_DS_TL	W_RTL_PROMO_D_TL
W_RTL_SEASON_DS	W_RTL_SEASON_D
W_RTL_SEASON_PHASE_IT_DS	W_RTL_SEASON_PHASE_IT_D
W_RTL_TNDR_TYPE_DS	W_RTL_TNDR_TYPE_D
W_STATUS_DS	W_STATUS_D

Initial Dimension Preprocessing

Module Name	LOAD_EXT_DIM_INITIAL_SI_ADHOC
Description	Converts the simplified dimension data loaded from CSV files into the internal staging area of Retail Insights, for use in the final dimension loads.
Dependencies	LOAD_DIM_INITIAL_CSV_ADHOC
Business Activity	Historical Data Load

Design Overview

This process applies all of the transformation scripts needed to take simplified interface data for dimensions (loaded from CSV files) and map it to the internal RI data model staging tables. The simplified interfaces are a one-to-many mapping to the legacy RI structures for dimensions, so this intermediate step is required to transform the incoming data and make it usable in RI. The primary benefit of this standalone process for transformations is to give you the opportunity to validate that all your data converts properly from the input file, before loading it into the final dimension tables. For example, the `PRODUCT.csv` file may write to 4 or more tables, and you can validate each step of the process has been applied correctly before loading it the rest of the way into RI. Improper formatting or missing values in the source data could cause unexpected results during this process, so it is useful to perform extra validation on the first loading a new dataset.

Key Tables Affected

Input Table	Output Table
W_PRODUCT_DTS	W_PROD_CAT_DHS
W_PRODUCT_DTS	W_PRODUCT_ATTR_DS
W_PRODUCT_DTS	W_PRODUCT_DS
W_PRODUCT_DTS	W_PRODUCT_DS_TL
W_PRODUCT_DTS	W_RTL_PRODUCT_BRAND_DS
W_PRODUCT_DTS	W_RTL_PRODUCT_BRAND_DS_TL

Input Table	Output Table
W_PRODUCT_DTS	W_RTL_IT_SUPPLIER_DS
W_PRODUCT_DTS	W_DOMAIN_MEMBER_DS_TL
W_INT_ORG_DTS	W_INT_ORG_DS
W_INT_ORG_DTS	W_INT_ORG_DS_TL
W_INT_ORG_DTS	W_INT_ORG_DHS
W_INT_ORG_DTS	W_DOMAIN_MEMBER_DS_TL
W_INT_ORG_DTS	W_RTL_CHANNEL_DS
W_INT_ORG_DTS	W_INT_ORG_ATTR_DS
W_EXCH_RATE_DTS	W_EXCH_RATE_GS
W_MCAL_PERIODS_DTS	W_MCAL_PERIOD_DS
W_SUPPLIER_DTS	W_PARTY_ATTR_DS
W_SUPPLIER_DTS	W_PARTY_ORG_DS
W_EMPLOYEE_DTS	W_EMPLOYEE_DS
W_PROD_LOC_ATTR_DTS	W_RTL_IT_LC_DS
W_INVENTORY_PRODUCT_ATTR_DTS	W_INVENTORY_PRODUCT_ATTR_DS
W_ATTR_DTS	W_RTL_PRODUCT_ATTR_DS
W_ATTR_DTS	W_RTL_PRODUCT_ATTR_DS_TL
W_ATTR_DTS	W_DOMAIN_MEMBER_DS_TL
W_ATTR_DTS	W_RTL_PRODUCT_COLOR_DS
W_PRODUCT_ATTR_DTS	W_RTL_ITEM_GRP1_DS
W_RTL_SEASON_PHASE_DTS	W_RTL_SEASON_DS
W_RTL_SEASON_PHASE_DTS	W_RTL_PHASE_DS
W_RTL_SEASON_PHASE_DTS	W_DOMAIN_MEMBER_DS_TL
W_RTL_SEASON_PHASE_IT_DTS	W_RTL_SEASON_PHASE_IT_DS
W_RTL_LOC_COMP_MTX_DTS	W_RTL_LOC_COMP_MTX_DS
W_RTL_CODE_DTS	W_RTL_CODE_DS
W_RTL_ITEM_GRP2_DTS	W_RTL_ITEM_GRP2_DS
W_DIFF_GROUP_DTS	W_RTL_DIFF_GRP_DS
W_DIFF_GROUP_DTS	W_RTL_DIFF_GRP_DS_TL
W_ADJUSTMENT_FTS	W_REASON_DS
W_ADJUSTMENT_FTS	W_DOMAIN_MEMBER_DS_TL
W_RTL_PROMO_EXT_DTS	W_RTL_PROMO_EXT_DS
W_ORDER_HEAD_FTS	W_RTL_PO_DETAILS_DS

Initial Dimension Staging

Module Name	LOAD_DIM_INITIAL_CSV_ADHOC
Description	Stages all of the dimension CSV files from the server for initial data loads into the database.

Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

This process looks for all CSV files for dimensions placed on the server by a history ZIP file upload and moves them into preprocessing tables in RI for use by the `LOAD_EXT_DIM_INITIAL_SI_ADHOC` process.

Key Tables Affected

Table	Usage
W_PRODUCT_DTS	File Input
W_INT_ORG_DTS	File Input
W_EXCH_RATE_DTS	File Input
W_MCAL_PERIODS_DTS	File Input
W_SUPPLIER_DTS	File Input
W_EMPLOYEE_DTS	File Input
W_PROD_LOC_ATTR_DTS	File Input
W_INVENTORY_PRODUCT_ATTR_DTS	File Input
W_ATTR_DTS	File Input
W_PRODUCT_ATTR_DTS	File Input
W_RTL_SEASON_PHASE_DTS	File Input
W_RTL_SEASON_PHASE_IT_DTS	File Input
W_RTL_LOC_COMP_MTX_DTS	File Input
W_RTL_CODE_DTS	File Input
W_RTL_ITEM_GRP2_DTS	File Input
W_DIFF_GROUP_DTS	File Input
W_ADJUSTMENT_FTS	File Input
W_RTL_PROMO_EXT_DTS	File Input
W_ORDER_HEAD_FTS	File Input

Initial Dimension Staging (Legacy)

Module Name	LOAD_DIM_INITIAL_STAGE_ADHOC
Description	Stages all of the dimension DAT files from the server for initial data loads into the database.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

This process looks for all legacy DAT files for dimensions placed on the server by a history ZIP file upload and move them into staging tables in RI for use by the `LOAD_DIM_INITIAL_ADHOC` process. You cannot use both the CSV and DAT staging processes for the same data, as they overwrite each other. However, you may use this process to load DAT files for an interface where a CSV file does not exist, such as `W_PARTY_PER_DS.dat`.

Key Tables Affected

Table	Usage
RA_SRC_CURR_PARAM_G	File Input
W_CODE_DS	File Input
W_DOMAIN_MEMBER_DS_TL	File Input
W_EMPLOYEE_DS	File Input
W_EXCH_RATE_GS	File Input
W_INT_ORG_ATTR_DS	File Input
W_INT_ORG_DHS	File Input
W_INT_ORG_DS	File Input
W_INT_ORG_DS_TL	File Input
W_PARTY_ATTR_DS	File Input
W_PARTY_ORG_DS	File Input
W_PARTY_PER_DS	File Input
W_PROD_CAT_DHS	File Input
W_PRODUCT_ATTR_DS	File Input
W_PRODUCT_DS	File Input
W_PRODUCT_DS_TL	File Input
W_REASON_DS	File Input
W_RTL_ALC_DETAILS_DS	File Input
W_RTL_BUYER_DS	File Input
W_RTL_CHANNEL_DS	File Input
W_RTL_CO_HEAD_DS	File Input
W_RTL_CO_LINE_DS	File Input
W_RTL_CO_SHIP_METHOD_DS	File Input
W_RTL_CO_SHIP_TYPE_DS	File Input
W_RTL_CODE_DS	File Input
W_RTL_COMP_STORE_DS	File Input
W_RTL_COUPON_DS	File Input
W_RTL_COUPON_DS_TL	File Input
W_RTL_DIFF_GRP_DS	File Input
W_RTL_DIFF_GRP_DS_TL	File Input

Table	Usage
W_RTL_DIFF_RNG_DS	File Input
W_RTL_DIFF_RNG_DS_TL	File Input
W_RTL_DISCOUNT_TYPE_DS	File Input
W_RTL_IT_SUPPLIER_DS	File Input
W_RTL_ITEM_GRP1_DS	File Input
W_RTL_LOC_STOCK_CNT_DS	File Input
W_RTL_ORG_FIN_DS	File Input
W_RTL_PARTY_PER_ATTR_DS	File Input
W_RTL_PHASE_DS	File Input
W_RTL_PO_DETAILS_DS	File Input
W_RTL_PRICE_CLR_IT_LC_DS	File Input
W_RTL_PROD_HIER_ATTR_LKP_DHS	File Input
W_RTL_PRODUCT_BRAND_DS	File Input
W_RTL_PRODUCT_BRAND_DS_TL	File Input
W_RTL_PROMO_CE_DS	File Input
W_RTL_PROMO_DS	File Input
W_RTL_PROMO_DS_TL	File Input
W_RTL_PROMO_EXT_DS	File Input
W_RTL_SEASON_DS	File Input
W_RTL_SEASON_PHASE_IT_DS	File Input
W_RTL_TNDR_TYPE_DS	File Input
W_STATUS_DS	File Input

Initial Inventory Seeding

Module Name	SEED_CSV_W_RTL_INV_IT_LC_DY_F_PROCESS_ADHOC
Description	Loads a full snapshot of inventory data from INVENTORY.csv to initialize the positional data before a nightly batch can be enabled.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Nightly Batch Preparation

Design Overview

The seeding load process for Inventory data accepts an input file at the item-location-date level using the file specification for INVENTORY.csv. It assumes the file has already been moved into place by the HIST_ZIP_FILE_LOAD_ADHOC process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day).

> **Note:**
>
> Seeding processes require a full snapshot of data for a single date, which covers all item/location combinations that should have a starting position for this fact. The seeding process must load data for the day before the nightly batch runs. Alternatively, you can include the full snapshots of data in your first nightly batch and skip the seeding steps. This causes the nightly batch to take a significantly longer time to execute, but avoids the manual load processes for all the positional facts.

Key Tables Affected

Table	Usage
W_RTL_INV_IT_LC_DY_FTS	File Input
W_RTL_INV_IT_LC_DY_FS	Staging
W_RTL_INV_IT_LC_G	Output
W_RTL_INV_IT_LC_DY_F	Output

Initial Inventory Seeding (Legacy)

Module Name	SEED_W_RTL_INV_IT_LC_DY_F_PROCESS_ADHOC
Description	Loads a full snapshot of inventory data from `W_RTL_INV_IT_LC_DY_FS.dat` to initialize the positional data before a nightly batch can be enabled.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Nightly Batch Preparation

Design Overview

The seeding load process for Inventory data accepts an input file at the item-location-date level using the file specification for `W_RTL_INV_IT_LC_DY_FS.dat`. It assumes the file has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file to RI's internal staging tables, then load it into the base fact (item/location/day).

> **Note:**
>
> Seeding processes require a full snapshot of data for a single date, which covers all item/location combinations that should have a starting position for this fact. The seeding process must load data for the day before the nightly batch runs. Alternatively, you can include the full snapshots of data in your very first nightly batch and skip the seeding steps. This causes the nightly batch to take a significantly longer time to execute but avoids the manual load processes for all the positional facts.

Key Tables Affected

Table	Usage
W_RTL_INV_IT_LC_DY_FS	File Input
W_RTL_INV_IT_LC_G	Output
W_RTL_INV_IT_LC_DY_F	Output

Initial Net Cost Seeding

Module Name	SEED_CSV_W_RTL_NCOST_IT_LC_DY_F_PROCESS_ADHOC
Description	Loads a full snapshot of net cost data from `COST.csv` to initialize the positional data before a nightly batch can be enabled.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Nightly Batch Preparation

Design Overview

The seeding load process for Net Cost data accepts an input file at the item-location-date-supplier level using the file specification for `COST.csv`. It assumes the file has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day). This process is only for the net cost; a separate process loads the base cost, if required.

> **Note:**
>
> Seeding processes require a full snapshot of data for a single date, which covers all item/location combinations that should have a starting position for this fact. The seeding process must load data for the day before the nightly batch runs. Alternatively, you can include the full snapshots of data in your very first nightly batch and skip the seeding steps. This causes the nightly batch to take a significantly longer time to execute, but avoids the manual load processes for all the positional facts.

Key Tables Affected

Table	Usage
W_COST_FTS	File Input
W_RTL_NCOST_IT_LC_DY_FS	Staging
W_RTL_NCOST_IT_LC_G	Output
W_RTL_NCOST_IT_LC_DY_F	Output

Initial Net Cost Seeding (Legacy)

Module Name	SEED_W_RTL_NCOST_IT_LC_DY_F_PROCESS_ADHOC
Description	Loads a full snapshot of net cost data from `W_RTL_NCOST_IT_LC_DY_FS.dat` to initialize the positional data before a nightly batch can be enabled.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Nightly Batch Preparation

Design Overview

The seeding load process for Net Cost data accepts an input file at the item-location-date-supplier level using the file specification for `W_RTL_NCOST_IT_LC_DY_FS.dat`. It assumes the file has already been moved into place using the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day).

> **Note:**
>
> Seeding processes require a full snapshot of data for a single date, which covers all item/location combinations that should have a starting position for this fact. The seeding process must load data for the day before the nightly batch runs. Alternatively, you can include the full snapshots of data in your very first nightly batch and skip the seeding steps. This causes the nightly batch to take a significantly longer time to execute, but avoids the manual load processes for all the positional facts.

Key Tables Affected

Table	Usage
W_RTL_NCOST_IT_LC_DY_FS	File Input
W_RTL_NCOST_IT_LC_G	Output
W_RTL_NCOST_IT_LC_DY_F	Output

Initial Price Seeding

Module Name	SEED_CSV_W_RTL_PRICE_IT_LC_DY_F_PROCESS_ADHOC
Description	Loads a full snapshot of price data from `PRICE.csv` to initialize the positional data before a nightly batch can be enabled.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Nightly Batch Preparation

Design Overview

The seeding load process for Price data accepts an input file at the item-location-date level using the file specification for `PRICE.csv`. It assumes the file has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day).

> **Note:**
>
> Seeding processes require a full snapshot of data for a single date, which covers all item/location combinations that should have a starting position for this fact. The seeding process must load data for the day before the nightly batch runs. Alternatively, you can include the full snapshots of data in your very first nightly batch and skip the seeding steps. This causes the nightly batch to take a significantly longer time to execute, but avoids the manual load processes for all the positional facts.

Key Tables Affected

Table	Usage
W_RTL_PRICE_IT_LC_DY_FTS	File Input
W_RTL_PRICE_IT_LC_DY_FS	Staging
W_RTL_PRICE_IT_LC_G	Output
W_RTL_PRICE_IT_LC_DY_F	Output

Initial Price Seeding (Legacy)

Module Name	SEED_W_RTL_PRICE_IT_LC_DY_F_PROCESS_ADHOC
Description	Loads a full snapshot of price data from `W_RTL_PRICE_IT_LC_DY_FS.dat` to initialize the positional data before a nightly batch can be enabled.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Nightly Batch Preparation

Design Overview

The seeding load process for Price data accepts an input file at the item-location-date level using the file specification for `W_RTL_PRICE_IT_LC_DY_FS.dat`. It assumes the file has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a staging table, then loads it into the base fact (item/location/day).

ORACLE

> **✎ Note:**
>
> Seeding processes require a full snapshot of data for a single date, which covers all item/location combinations that should have a starting position for this fact. The seeding process must load data for the day before the nightly batch runs. Alternatively, you can include the full snapshots of data in your very first nightly batch and skip the seeding steps. This causes the nightly batch to take a significantly longer time to execute but avoids the manual load processes for all the positional facts.

Key Tables Affected

Table	Usage
W_RTL_PRICE_IT_LC_DY_FS	File Input
W_RTL_PRICE_IT_LC_G	Output
W_RTL_PRICE_IT_LC_DY_F	Output

Initial Purchase Order Seeding

Module Name	SEED_CSV_W_RTL_PO_ONORD_IT_LC_DY_F_PROCESS_ADHOC
Description	Loads a full snapshot of purchase order data from `ORDER_HEAD.csv` and `ORDER_DETAIL.csv` to initialize the positional data before a nightly batch can be enabled.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Nightly Batch Preparation

Design Overview

The seeding load process for Purchase Order data accepts two input files at the order header and order detail levels using the file specifications for `ORDER_HEAD.csv` and `ORDER_DETAIL.csv`. It assumes the files have already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the files into preprocessing tables in the database, transforms them to RI's internal staging tables, then loads them into the base dimension and facts. The dimension is loaded first to support loading the fact table against those foreign keys.

> ✏ **Note:**
>
> Seeding processes require a full snapshot of data for a single date, which covers all purchase orders and item/location combinations that should have a starting position for this fact. The seeding process must load data for the day before the nightly batch runs. Alternatively, you can include the full snapshots of data in your very first nightly batch and skip the seeding steps. This causes the nightly batch to take a significantly longer time to execute but avoids the manual load processes for all the positional facts.

Key Tables Affected

Table	Usage
W_ORDER_HEAD_FTS	File Input
W_RTL_PO_DETAILS_DS	Staging
W_RTL_PO_DETAILS_D	Output
W_ORDER_DETAIL_FTS	File Input
W_RTL_PO_ONORD_IT_LC_DY_FS	Staging
W_RTL_PO_ONORD_IT_LC_DY_F	Output

Initial Purchase Order Seeding (Legacy)

Module Name	SEED_W_RTL_PO_ONORD_IT_LC_DY_F_PROCESS_ADHOC
Description	Loads a full snapshot of purchase order data from `W_RTL_PO_ONORD_IT_LC_DY_FS.dat` to initialize the positional data before a nightly batch can be enabled.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Nightly Batch Preparation

Design Overview

The seeding load process for Purchase Order fact data accepts an input file at the item-location-date level using the file specification for `W_RTL_PO_ONORD_IT_LC_DY_FS.dat`. It assumes the file has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a staging table, then loads it into the base fact (item/location/day). It assumes the dimension has already been loaded separately using the initial dimension loads.

> **✎ Note:**
>
> Seeding processes require a full snapshot of data for a single date, which covers all item/location combinations that should have a starting position for this fact. The seeding process must load data for the day before the nightly batch runs. Alternatively, you can include the full snapshots of data in your very first nightly batch and skip the seeding steps. This causes the nightly batch to take a significantly longer time to execute but avoids the manual load processes for all the positional facts

Key Tables Affected

Table	Usage
W_RTL_PO_ONORD_IT_LC_DY_FS	File Input
W_RTL_PO_ONORD_IT_LC_DY_F	Output

Intercompany Margin History Load

Module Name	HIST_CSV_ICMARGIN_LOAD_ADHOC
Description	Loads the `IC_MARGIN.csv` file into RI and populates key data tables used to integrate with other systems for history data.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Nightly Batch Preparation

Design Overview

The history load process for Intercompany Margin transactions accepts an input file at the item/location/day level using the file specification for `IC_MARGIN.csv`. It assumes the file has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day) as well as the week aggregate used for integrations (item/location/week).

Key Tables Affected

Table	Usage
W_RTL_ICM_IT_LC_DY_FTS	File Input
W_RTL_ICM_IT_LC_DY_FS	Staging
W_RTL_ICM_IT_LC_DY_F	Output (Base Fact)
W_RTL_ICM_IT_LC_WK_A	Output (Aggregate)

Inventory History Load

Module Name	HIST_INV_LOAD_ADHOC
Description	Processes any staged inventory history data for end-of-week snapshots, starting from the last processed week.
Dependencies	HIST_STG_CSV_INV_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

The inventory history load process supports loading of end-of-week inventory snapshots over a long period of time to populate RI with historical data. It requires the inventory data to already be staged into the database by one of the available staging processes. Multiple weeks of inventory can be provided in a single file, though it is recommended to not load more than one month at a time unless the volumes are low. Every record in the data <u>must</u> be for a week-ending date; other dates in the file will not work using this process.

The `C_HIST_LOAD_STATUS` configuration table controls the actions taken by the process. Before running the process for the first time, you must set up this table for the following:

- Set the history load date to be the very latest date you expect to load history for (this can be changed later if needed to load more weeks)

- Disable any aggregate tables you do not wish to populate. When loading data only for AI Foundation, you only need the base fact (`W_RTL_INV_IT_LC_DY_F`) and week aggregate (`W_RTL_INV_IT_LC_WK_A`). For RI, all tables should be enabled and loaded.

Once setup is complete, begin processing files from the <u>earliest</u> week-ending date you plan to load. You must start from the beginning of the history and load data sequentially. You cannot load data out of order and you cannot load the same week multiple times without first erasing the data from your database. After a week is loaded successfully, the `C_HIST_LOAD_STATUS` records are updated with the most recent load status and date.

Key Tables Affected

Table	Usage
C_HIST_LOAD_STATUS	Configuration
W_RTL_INV_IT_LC_DY_FS	Input
W_RTL_INV_IT_LC_DY_F	Output
W_RTL_INV_IT_LC_G	Output
W_RTL_INV_IT_LC_GMH_A	Output
W_RTL_INV_IT_LC_WK_A	Output
W_RTL_INV_IT_DY_A	Output
W_RTL_INV_IT_WK_A	Output
W_RTL_INV_SC_LC_DY_A	Output
W_RTL_INV_CL_LC_DY_A	Output
W_RTL_INV_DP_LC_DY_A	Output

W_RTL_INV_SC_LC_DY_CUR_A	Output
W_RTL_INV_SC_DY_A	Output
W_RTL_INV_SC_DY_CUR_A	Output
W_RTL_INV_SC_LC_WK_A	Output
W_RTL_INV_CL_LC_WK_A	Output
W_RTL_INV_DP_LC_WK_A	Output
W_RTL_INV_SC_LC_WK_CUR_A	Output
W_RTL_INV_SC_WK_A	Output
W_RTL_INV_SC_WK_CUR_A	Output

Inventory History Staging

Module Name	HIST_STG_CSV_INV_LOAD_ADHOC
Description	Stages the INVENTORY.csv file for the ad hoc inventory load programs.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

This process looks for the INVENTORY.csv file placed on the server by a history zip file upload, move it into a preprocessing table in RI, and transform it for use in the HIST_INV_LOAD_ADHOC process.

> ✎ **Note:**
>
> The inventory file used for history data must contain only week-ending dates and must be full, weekly snapshots of data.

Key Tables Affected

Table	Usage
W_RTL_INV_IT_LC_DY_FTS	File Input
W_RTL_INV_IT_LC_DY_FS	Output

Inventory History Staging (Legacy)

Module Name	HIST_STG_INV_LOAD_ADHOC
Description	Stages the W_RTL_INV_IT_LC_DY_FS.dat file for the ad hoc inventory load programs.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC

Business Activity	Historical Data Load

Design Overview

This process looks for the `W_RTL_INV_IT_LC_DY_FS.dat` file placed on the server by a history ZIP file upload and loads it for use by the `HIST_INV_LOAD_ADHOC` process.

> ✏️ **Note:**
>
> The inventory file used for history data must contain only week-ending dates and must be full, weekly snapshots of data.

Key Tables Affected

Table	Usage
W_RTL_INV_IT_LC_DY_FS	File Input

Inventory Reclass History Load

Module Name	HIST_CSV_INVRECLASS_LOAD_ADHOC
Description	Loads the `INV_RECLASS.csv` file into RI and populates key data tables used to integrate with other systems for history data.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

The history load process for Inventory Reclass transactions accepts an input file at the item/location/day level using the file specification for `INV_RECLASS.csv`. It assumes the file has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day) as well as the week aggregate used for integrations (item/location/week).

Key Tables Affected

Table	Usage
W_RTL_INVRECLASS_IT_LC_DY_FTS	File Input
W_RTL_INVRECLASS_IT_LC_DY_FS	Staging
W_RTL_INVRECLASS_IT_LC_DY_F	Output (Base Fact)
W_RTL_INVRECLASS_IT_LC_WK_A	Output (Aggregate)

Inventory Selling Date Seeding

Module Name	LOAD_W_RTL_INV_IT_LC_G_FIRST_SOLD_DT_ADHOC
Description	Calculates the initial value of First Sold Date for all item/locations in inventory, based on sales history data.
Dependencies	SEED_CSV_W_RTL_INV_IT_LC_DY_F_PROCESS_ADHOC
Business Activity	Historical Data Load

Design Overview

This process populates the fields `W_RTL_INV_IT_LC_G.FIRST_SOLD_DT` and `LAST_SOLD_DT` with values, using your historical sales data to calculate the first time each item/location with stock on hand was sold. This process should only run after all inventory and sales history is completely loaded and you are ready to begin nightly batches. If this process does not run, then all item/locations will start with a first/last selling date of the first transaction to occur on it in nightly batch runs. These date values are used by the AI Foundation Cloud Services (Pricing and Markdown Optimization) as an input to determine item lifecycles from the history data in RI.

Key Tables Affected

Table	Usage
W_RTL_SLS_TRX_IT_LC_DY_F	Input
W_RTL_INV_IT_LC_G	Output

Markdown History Load

Module Name	HIST_CSV_MARKDOWN_LOAD_ADHOC
Description	Loads the `MARKDOWN.csv` file into RI and populates key data tables used to integrate with other systems for history data.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

The history load process for Markdown transactions accepts an input file at the item/location/day level using the file specification for `MARKDOWN.csv`. It assumes the file has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day) as well as the week aggregate used for integrations (item/location/week).

Key Tables Affected

Table	Usage
W_MARKDOWN_FTS	File Input
W_RTL_MKDN_IT_LC_DY_FS	Staging
W_RTL_MKDN_IT_LC_DY_F	Output (Base Fact)
W_RTL_MKDN_IT_LC_WK_A	Output (Aggregate)

Nightly Batch Status Cleanup

Module Name	C_LOAD_DATES_CLEANUP_ADHOC
Description	Erases the execution status of nightly batch programs. This is required to run a nightly process outside of a batch.
Dependencies	None
Business Activity	Batch Administration

Design Overview

This process erases records from the `C_LOAD_DATES` database table. Any time a job runs as part of the nightly batch, or a job runs which is included in both nightly and ad hoc processing, a status record is inserted to `C_LOAD_DATES`. The job is then blocked from executing again while this record exists, as a safety measure when restarting batch processes that failed midway through execution. During initial dimension loads, you may need to execute the same jobs multiple times to work through file or data issues. In that case, you may execute this process before each run to clear the status of prior runs from the database.

> **Note:**
>
> This process should only run during history and initial data loads or at the guidance of Oracle Support. It should not be run during regular nightly batch processing. Clearing `C_LOAD_DATES` while the batch is running normally could cause data corruption, as it would allow the same jobs to run multiple times for the same business date.

Key Tables Affected

Table	Usage
C_LOAD_DATES	Delete

Plan Data Integration

Module Name	LOAD_PLANNING1_DATA_ADHOC
	LOAD_PLANNING2_DATA_ADHOC
	LOAD_PLANNING3_DATA_ADHOC
	LOAD_PLANNING4_DATA_ADHOC
Description	Extracts data from the MFP Plan Export interfaces to RI's internal planning tables.
Dependencies	CLEANUP_C_LOAD_DATES_PLANNING_ADHOC
Business Activity	RI Integrations

Design Overview

This set of processes moves Merchandise Financial Planning (MFP) export data from the data exchange (RDX) layer to internal RI staging tables, then triggers the RI load programs for planning data. Each process contains the end-to-end flow of data for a single interface. Use these processes to perform integration testing and plan data validations during an RI and MFP implementation, or to trigger an on-demand refresh of plan data in RI outside the normal batch cycle. If you run these on the same day as a normal batch run, or you run them multiple times, you must run the cleanup process shown in the dependencies prior to each run.

Key Tables Affected

Table	Usage
MFP_PLAN1_EXP	Input
W_RTL_PLAN1_PROD1_LC1_T1_FS	Staging
W_RTL_PLAN1_PROD1_LC1_T1_F	Output
MFP_PLAN2_EXP	Input
W_RTL_PLAN2_PROD2_LC2_T2_FS	Staging
W_RTL_PLAN2_PROD2_LC2_T2_F	Output
MFP_PLAN3_EXP	Input
W_RTL_PLAN3_PROD3_LC3_T3_FS	Staging
W_RTL_PLAN3_PROD3_LC3_T3_F	Output
MFP_PLAN4_EXP	Input
W_RTL_PLAN4_PROD4_LC4_T4_FS	Staging
W_RTL_PLAN4_PROD4_LC4_T4_F	Output

Planning Dimension Export

Module Name	LOAD_PDS_DIMENSION_PROCESS_ADHOC
Description	Exports all supported dimensions from RI to the data exchange schema for Planning.

Dependencies	LOAD_DIM_INITIAL_ADHOC
Business Activity	RI Integrations

Design Overview

This process runs all the planning data schema dimension exports from RI to the data exchange layer, where PDS batch processes can pick up and load the data the rest of the way. Each time the exports run, the data is truncated and inserted as full snapshots. Planning exports do not support incremental or delta extracts for dimensions. RI applies various filters and criteria to the export data to align with Planning Data Schema requirements for dimensions, as described in the *RAP Implementation Guide*. RI only exports specific columns from each dimension, based on the downstream application needs. Review the PDS integration tables in detail to understand which data will be exported.

Key Tables Affected

Input Table	Output Table
W_PRODUCT_D	W_PDS_PRODUCT_D
W_PRODUCT_D_TL	W_PDS_PRODUCT_D
W_PROD_CAT_DH	W_PDS_PRODUCT_D
W_PRODUCT_ATTR_D	W_PDS_PRODUCT_D
W_DOMAIN_MEMBER_LKP_TL	W_PDS_PRODUCT_D
W_INT_ORG_D	W_PDS_ORGANIZATION_D
W_INT_ORG_D_TL	W_PDS_ORGANIZATION_D
W_INT_ORG_DH	W_PDS_ORGANIZATION_D
W_DOMAIN_MEMBER_LKP_TL	W_PDS_ORGANIZATION_D
W_INT_ORG_ATTR_D	W_PDS_ORGANIZATION_D
W_MCAL_DAY_D	W_PDS_CALENDAR_D
W_EXCH_RATE_G	W_PDS_EXCH_RATE_G
W_RTL_ITEM_GRP1_D	W_PDS_PRODUCT_ATTR_D
W_DOMAIN_MEMBER_LKP_TL	W_PDS_PRODUCT_ATTR_D
W_RTL_PRODUCT_ATTR_D	W_PDS_UDA_D
W_DOMAIN_MEMBER_LKP_TL	W_PDS_UDA_D
W_RTL_PRODUCT_ATTR_D	W_PDS_DIFF_D
W_RTL_PRODUCT_ATTR_D_TL	W_PDS_DIFF_D

Planning Fact Export

Module Name	LOAD_PDS_FACT_PROCESS_ADHOC
Description	Exports all supported facts from RI to the data exchange schema for Planning.

Dependencies	HIST_SALES_LOAD_ADHOC
	HIST_INV_LOAD_ADHOC
	HIST_CSV_ADJUSTMENTS_LOAD_ADHOC
	HIST_CSV_INVRECEIPTS_LOAD_ADHOC
	HIST_CSV_MARKDOWN_LOAD_ADHOC
	HIST_CSV_INVRTV_LOAD_ADHOC
	HIST_CSV_TRANSFER_LOAD_ADHOC
	HIST_CSV_DEAL_INCOME_LOAD_ADHOC
	HIST_CSV_ICMARGIN_LOAD_ADHOC
	HIST_CSV_INVRECLASS_LOAD_ADHOC
Business Activity	RI Integrations

Design Overview

This process runs all the planning data schema fact exports from RI to the data exchange layer, where PDS batch processes pick up and load the data the rest of the way. Each run of these jobs inserts to the target tables with a new `RUN_ID`. Old runs are preserved for a configurable period of time (such as 7 days) to ensure PDS has adequate time to retrieve the data before it is erased. All fact exports are incremental and send only the current week's data based on when it was posted into RI. This means the exports include all back-posted transaction data regardless of the transaction date, as long as it was posted to RI in the current fiscal week.

The range of dates exported by this process is tracked and configured from the table `C_SOURCE_CDC`. This table can be edited from the Control & Tactical Center to alter the range of dates exported in one batch execution, such as when you are sending historical data to MFP, or when you need to refresh the PDS data for more than a week. The table is automatically updated after every run to reflect the most recent export dates. The next export begin from the last date/time used.

Key Tables Affected

Input Table	Output Table
W_RTL_SLS_IT_LC_WK_A	W_PDS_SLS_IT_LC_WK_A
W_RTL_SLS_IT_LC_WK_A	W_PDS_GRS_SLS_IT_LC_WK_A
W_RTL_SLSWF_IT_LC_WK_A	W_PDS_SLSWF_IT_LC_WK_A
W_RTL_INV_IT_LC_WK_A	W_PDS_INV_IT_LC_WK_A
W_RTL_PO_ONORD_IT_LC_DY_F	W_PDS_PO_ONORD_IT_LC_WK_A
W_RTL_MKDN_IT_LC_WK_A	W_PDS_MKDN_IT_LC_WK_A
W_RTL_INVADJ_IT_LC_WK_A	W_PDS_INVADJ_IT_LC_WK_A
W_RTL_INVRC_IT_LC_WK_A	W_PDS_INVRC_IT_LC_WK_A
W_RTL_INVTSF_IT_LC_WK_A	W_PDS_INVTSF_IT_LC_WK_A
W_RTL_INVRTV_IT_LC_WK_A	W_PDS_INVRTV_IT_LC_WK_A
W_RTL_INVRECLASS_IT_LC_WK_A	W_PDS_INVRECLASS_IT_LC_WK_A
W_RTL_DEALINC_IT_LC_WK_A	W_PDS_DEALINC_IT_LC_WK_A
W_RTL_ICM_IT_LC_WK_A	W_PDS_ICM_IT_LC_WK_A

Planning Initial Inventory Export

Module Name	LOAD_PDS_FACT_INITIAL_PROCESS_ADHOC
Description	Exports a full snapshot of historical inventory to the data exchange schema for Planning.
Dependencies	HIST_INV_LOAD_ADHOC
Business Activity	RI Integrations

Design Overview

This process exports inventory history from RI to Planning. The base inventory extract for PDS only sends the current week's inventory, as the data is positional in RI and the current week reflects all current values on the fact. This process can send a range of weeks at one time by configuring the start date and end date in C_SOURCE_CDC for this interface. All weeks of data are written for a single Run ID in the output table. Running the PDS import process consumes the entire range of data into their inventory facts.

Key Tables Affected

Table	Usage
C_SOURCE_CDC	Configuration
W_RTL_INV_IT_LC_WK_A	Input
W_PDS_INV_IT_LC_WK_A	Output

Planning Load Cleanup

Module Name	CLEANUP_C_LOAD_DATES_PLANNING_ADHOC
Description	Erases the execution status of planning batch programs. This is required to run a program multiple times for the same business date.
Dependencies	None
Business Activity	RI Integrations

Design Overview

This process erases records from the C_LOAD_DATES database table. Any time a job runs as part of the nightly batch, or a job is included in both nightly and ad hoc processing, a status record is inserted to C_LOAD_DATES. The job is then blocked from executing again while this record exists, as a safety measure when restarting batch processes that failed midway through execution. During initial planning integration loads, you may need to execute the same jobs multiple times to work through file or data issues. In that case, you may execute this process before each run to clear the status of prior runs from the database.

> **✎ Note:**
>
> This process should only run during history and initial data loads, or at the guidance of Oracle Support. It should not run during regular nightly batch processing. Clearing `C_LOAD_DATES` when the batch is running normally could cause data corruption, as it would allow the same jobs to run multiple times for the same business date.

Key Tables Affected

Table	Usage
C_LOAD_DATES	Delete

POS Sales Integration

Module Name	LOAD_POSLOG_DATA_ADHOC
Description	Integrates data from Xstore, received through the POSLOG broadcaster services, into the RI data model.
Dependencies	None
Business Activity	RI Integrations

Design Overview

Retail Insights supports loading intraday sales transactions from Xstore's string-based XML receiver API. The data loaded by this method is specifically for reporting today's sales before the end-of-day batch processes the full snapshot of audited sales transactions. The sales data from Xstore is not used as a primary source of sales history in Retail Insights, as the system was designed around the concept of a Sales Audit system being used prior to data coming into the data warehouse.

The data first comes to the Retail AI Foundation Cloud Services from Xstore's web service API. The API is configured as part of the AI Foundation Cloud Services, but is used by Retail Insights to get the raw XML POSLOGs into the database for transformation to the RI data model. This process can then move the data from AI Foundation to RI staging tables, and from there to RI's internal data model for BI reports. Refer to the *RI Implementation Guide* for additional details.

Key Tables Affected

Table	Usage
W_RTL_POSLOG_XML_G	Input
W_RTL_SLS_POS_IT_LC_DY_FS	Staging
W_RTL_SLS_POS_IT_LC_DY_F	Output

Price History Load

Module Name	HIST_CSV_PRICE_LOAD_ADHOC
Description	Loads the `PRICE.csv` file into RI and populates key data tables used to integrate with other systems for history data.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

The price history load process supports loading of price information over a long period of time to populate RI with historical data. This process both stages the `PRICE.csv` file into the database and processes it into RI. Multiple weeks of pricing data can be provided in a single file, though it is recommended not to load more than one month at a time, unless the volumes are low. Pricing data <u>must</u> start with a full snapshot of all item/locations on the earliest day in history that you will be loading. This can be loaded by itself to validate the file is formatted and the data is correct. From then on, you can provide only the price change events on the dates that they occur (such as regular and markdown price changes). The price history load will iterate through the provided files one day at a time and load the available price change events for each date in order.

The `C_HIST_LOAD_STATUS` configuration table determines the actions taken by the process. Before running the process for the first time, you must set up this table for the history load date to be the very latest date you expect to load history for (this can be changed later if needed to load more weeks). Once that setup is complete, you can begin processing files from the <u>earliest</u> date you plan to load. You must start from the beginning of history and load sequentially. You cannot load data out of order, and you cannot load the same date multiple times without first erasing the data from the database. After a date is loaded successfully, the `C_HIST_LOAD_STATUS` records are updated with the most recent load status and date.

Key Tables Affected

Table	Usage
W_RTL_PRICE_IT_LC_DY_FTS	File Input
W_RTL_PRICE_IT_LC_DY_FS	Staging
W_RTL_PRICE_IT_LC_DY_F	Output
W_RTL_PRICE_IT_LC_G	Output

Price History Load (Legacy)

Module Name	HIST_PRICE_LOAD_ADHOC
Description	Stages and loads the `W_RTL_PRICE_IT_LC_DY_FS.dat` file for pricing history.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

The price history load process supports loading of price information over a long period of time to populate RI with historical data. This process stages the `W_RTL_PRICE_IT_LC_DY_FS.dat` file into the database and processes it into RI. Multiple weeks of pricing data can be provided in a single file, though it is recommended not to load more than one month at a time unless the volumes are low. Pricing data must start with a full snapshot of all item/locations on the earliest day in the history that you are loading. This can be loaded by itself to validate the file is formatted and the data is correct. From then on, you can provide only the price change events on the dates that they occur (such as regular and markdown price changes). The price history load iterates through the provided files one day at a time and loads the available price change events for each date in order.

The actions taken by the process are guided by a configuration table `C_HIST_LOAD_STATUS`. Before running the process for the first time, you must set up this table for the history load date to be the very latest date you expect to load history for (this can be changed later if needed to load more weeks). Once that setup is complete, you can begin processing files from the earliest date you plan to load. You must start from the beginning of history and load sequentially. You cannot load data out of order and you cannot load the same date multiple times without first erasing the data from your database. After a date is loaded successfully, the `C_HIST_LOAD_STATUS` records are updated with the most recent load status and date.

Key Tables Affected

Table	Usage
W_RTL_PRICE_IT_LC_DY_FS	Staging
W_RTL_PRICE_IT_LC_DY_F	Output
W_RTL_PRICE_IT_LC_G	Output

Receipts History Load

Module Name	HIST_CSV_INVRECEIPTS_LOAD_ADHOC
Description	Loads the `RECEIPT.csv` file into RI and populates key data tables used to integrate with other systems for history data.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

The history load process for Inventory Receipt transactions accepts an input file at the item/location/day level using the file specification for `RECEIPT.csv`. It assumes the file has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day) as well as the week aggregate used for integrations (item/location/week).

Key Tables Affected

Table	Usage
W_RECEIPT_FTS	File Input
W_RTL_INVRC_IT_LC_DY_FS	Staging
W_RTL_INVRC_IT_LC_DY_F	Output (Base Fact)
W_RTL_INVRC_IT_LC_WK_A	Output (Aggregate)

Rejected Record Analysis

Module Name	W_RTL_REJECT_DIMENSION_TMP_ADHOC
Description	Analyses rejected records in the pricing and inventory position facts for any known causes of rejection, such as missing dimension keys for the records, and outputs a summary for review.
Dependencies	None
Business Activity	Historical Data Load

Design Overview

The rejected record analysis ad hoc process provides a set of queries comparing rejected data to all related dimensional tables. If any dimension keys are found on the rejected data but not in the related tables, a summary of the comparison is output to a database table for review. This tool can help debug invalid input data so it can be corrected and reprocessed. The ad hoc job currently runs for the Sales Transaction, Inventory Position, and Pricing facts, which are the most common history loads performed. The job is run automatically for Inventory and Price loads since they will fail if any records are rejected. The modules enabled for the job are listed in the configuration table `W_RTL_REJECT_DIMLKUP_TMP`. The rejected dimension keys are output to `W_RTL_REJECT_DIMENSION_TMP`.

Key Tables Affected

Table	Usage
W_RTL_REJECT_DIMLKUP_TMP	Configuration
W_RTL_REJECT_DIMENSION_TMP	Output

Rejected Record Cleanup

Module Name	REJECT_DATA_CLEANUP_ADHOC
Description	Purges rejected records from certain `E$` tables and populates a list of invalid dimension keys present on the purged data. The invalid keys will be ignored if a related fact history load process is re-run after failing due to these rejections.
Dependencies	None

Business Activity   Historical Data Load

Design Overview

The rejected record cleanup ad hoc process provides a way to clear out rejected data for positional fact history loads (currently inventory and price) that are blocked by having any rejections. The data is erased from the `E$` tables and any invalid keys that do not have matching dimensions are written to the `C_DISCARD_DIMM` output table. If you then re-run the failed history job from POM, the job will ignore all of the discarded dimension keys and proceed to load the rest of the data file for the current day/week of processing. It is important to note that once you discard positional data in this manner, you cannot reload it later on: you are declaring the data as unwanted/unusable. If you instead want to reload your data file with corrected records, you would not re-run your current history load job. You would go back and reload dimension and fact files as needed and start a fresh job run.

Key Tables Affected

Table	Usage
E$_W_RTL_INV_IT_LC_DY_TMP1	Input
E$_W_RTL_PRICE_IT_LC_DP_TMP	Input
C_DISCARD_DIMM	Output

RTV History Load

Module Name	HIST_CSV_INVRTV_LOAD_ADHOC
Description	Loads the `RTV.csv` file into RI and populates key data tables used to integrate with other systems for history data.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

The history load process for Inventory Returns to Vendor (RTV) transactions accept an input file at the item/location/day level using the file specification for `RTV.csv`. It assumes the file has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day) as well as the week aggregate used for integrations (item/location/week).

Key Tables Affected

Table	Usage
W_RTL_INVRTV_IT_LC_DY_FTS	File Input

Table	Usage
W_RTL_INVRTV_IT_LC_DY_FS	Staging
W_RTL_INVRTV_IT_LC_DY_F	Output (Base Fact)
W_RTL_INVRTV_IT_LC_WK_A	Output (Aggregate)

RTV History Load (Legacy)

Module Name	HIST_INVRTV_LOAD_ADHOC
Description	Stages and loads the `W_RTL_INVRTV_IT_LC_DY_FS.dat` file for return-to-vendor history.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

The history load process for Inventory Returns to Vendor (RTV) transactions accepts an input file at the item/location/day level using the file specification for `W_RTL_INVRTV_IT_LC_DY_FS.dat`. It assumes the file has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process will import the file into RI's internal staging tables and then load it into the base fact (item/location/day) as well as the week aggregate used for integrations (item/location/week).

Key Tables Affected

Table	Usage
W_RTL_INVRTV_IT_LC_DY_FS	Staging
W_RTL_INVRTV_IT_LC_DY_F	Output (Base Fact)
W_RTL_INVRTV_IT_LC_WK_A	Output (Aggregate)

Sales History Load

Module Name	HIST_SALES_LOAD_ADHOC
Description	Processes any staged sales history data and runs all aggregation programs for a specified history range.
Dependencies	HIST_STG_CSV_SALES_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

The sales history load process supports loading of sales transaction data over a long period of time to populate RI with historical data. It requires the sales data to already be staged into the database using one of the available staging processes. Multiple weeks of sales can be provided in a single file, though it is recommended to not load more than one month at a time

unless the volumes are low. This process populates all sales tables in RI, both for integration and BI reporting purposes. If you are not using RI for reporting, disable the aggregation table programs in POM (except the `IT_LC_WK_A` aggregate) before running the process.

Key Tables Affected

Table	Usage
W_RTL_SLS_TRX_IT_LC_DY_FS	Input
W_RTL_SLS_TRX_IT_LC_DY_F	Output (Base Fact)
W_RTL_SLS_IT_LC_WK_A	Aggregate (for integrations)
W_RTL_SLS_IT_LC_DY_A	Aggregate (for BI reporting)
W_RTL_SLS_IT_LC_GMH_A	Aggregate (for BI reporting)
W_RTL_SLS_SC_LC_DY_A	Aggregate (for BI reporting)
W_RTL_SLS_SC_LC_WK_A	Aggregate (for BI reporting)
W_RTL_SLS_CL_LC_DY_A	Aggregate (for BI reporting)
W_RTL_SLS_CL_LC_WK_A	Aggregate (for BI reporting)
W_RTL_SLS_DP_LC_DY_A	Aggregate (for BI reporting)
W_RTL_SLS_DP_LC_WK_A	Aggregate (for BI reporting)
W_RTL_SLS_IT_DY_A	Aggregate (for BI reporting)
W_RTL_SLS_IT_WK_A	Aggregate (for BI reporting)
W_RTL_SLS_SC_DY_A	Aggregate (for BI reporting)
W_RTL_SLS_SC_WK_A	Aggregate (for BI reporting)
W_RTL_SLS_LC_DY_A	Aggregate (for BI reporting)
W_RTL_SLS_LC_WK_A	Aggregate (for BI reporting)
W_RTL_SLS_IT_LC_DY_SN_A	Aggregate (for BI reporting)
W_RTL_SLS_IT_LC_WK_SN_A	Aggregate (for BI reporting)
W_RTL_SLS_IT_DY_SN_A	Aggregate (for BI reporting)
W_RTL_SLS_IT_WK_SN_A	Aggregate (for BI reporting)
W_RTL_SLS_SC_LC_DY_CUR_A	Aggregate (for BI reporting)
W_RTL_SLS_SC_LC_WK_CUR_A	Aggregate (for BI reporting)
W_RTL_SLS_CL_LC_DY_CUR_A	Aggregate (for BI reporting)
W_RTL_SLS_DP_LC_DY_CUR_A	Aggregate (for BI reporting)
W_RTL_SLS_CL_LC_WK_CUR_A	Aggregate (for BI reporting)
W_RTL_SLS_DP_LC_WK_CUR_A	Aggregate (for BI reporting)
W_RTL_SLS_SC_DY_CUR_A	Aggregate (for BI reporting)
W_RTL_SLS_CL_DY_CUR_A	Aggregate (for BI reporting)
W_RTL_SLS_DP_DY_CUR_A	Aggregate (for BI reporting)
W_RTL_SLS_SC_WK_CUR_A	Aggregate (for BI reporting)
W_RTL_SLS_CL_WK_CUR_A	Aggregate (for BI reporting)
W_RTL_SLS_DP_WK_CUR_A	Aggregate (for BI reporting)
W_EMPLOYEE_D	Supporting Dimension (for BI reporting)

Table	Usage
W_PARTY_PER_D	Supporting Dimension (for BI reporting)
W_RTL_CO_HEAD_D	Supporting Dimension (for BI reporting)
W_RTL_CO_LINE_D	Supporting Dimension (for BI reporting)

Sales History Staging

Module Name	HIST_STG_CSV_SALES_LOAD_ADHOC
Description	Stages the SALES.csv file for the ad hoc sales load programs.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

This process looks for the SALES.csv file placed on the server by a history ZIP file upload, moves it into a preprocessing table in RI, and transforms it for use by the HIST_SALES_LOAD_ADHOC process.

Key Tables Affected

Table	Usage
W_RTL_SLS_TRX_IT_LC_DY_FTS	File Input
W_RTL_SLS_TRX_IT_LC_DY_FS	Output

Sales History Staging (Legacy)

Module Name	HIST_STG_SALES_LOAD_ADHOC
Description	Stages the W_RTL_SLS_TRX_IT_LC_DY_FS.dat file for the ad hoc sales load programs.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

This process looks for the W_RTL_SLS_TRX_IT_LC_DY_FS.dat file placed on the server by a history ZIP file upload and loads it for use in the HIST_SALES_LOAD_ADHOC process.

Key Tables Affected

Table	Usage
W_RTL_SLS_TRX_IT_LC_DY_FS	File Input

Table Partitioning

Module Name	CREATE_PARTITION_ADHOC
Description	Uses the provided range of dates and the loaded calendar information to generate table partitions across the RI data model.
Dependencies	CALENDAR_LOAD_ADHOC
Business Activity	Initial System Setup

Design Overview

This process must be used after the Calendar load is complete to partition all of your database tables. Tables in Retail Insights are partitioned dynamically based on your fiscal calendar using the days and weeks defined in `W_MCAL_DAY_D` and `W_MCAL_WEEK_D`. This type of partitioning provides optimal performance in BI reporting, where the SQL queries can prune the selected partitions to only those that hold data for your time-based filters and attributes. Without this partitioning in place, batch programs will not insert data into the expected partitions, some programs could fail to load data at all, and BI reporting will have very poor performance.

This process can be run repeatedly to ensure all partitions are created. Each time it is run, it will resume from where it left off, if any partitions still need to be added to the data model. If you have run the process several times and it is now completing in under a minute, then it is no long recreating any new partitions. The functional areas being partitioned should be reviewed in the table `C_MODULE_ARTIFACT`. All tables should be enabled for partitioning, except for tables that have `PLAN` in their naming structure.

Key Tables Affected

Table	Usage
W_MCAL_DAY_D	Input
W_MCAL_WEEK_D	Input
C_ODI_PARAM	Input

Transfer History Load

Module Name	HIST_CSV_TRANSFER_LOAD_ADHOC
Description	Loads the `TRANSFER.csv` file into RI and populates key data tables used to integrate with other systems for history data.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

The history load process for Inventory Transfer transactions accepts an input file at the item/location/day level using the file specification for `TRANSFER.csv`. It assumes the file

has already been moved into place by the `HIST_ZIP_FILE_LOAD_ADHOC` process. This process imports the file into a preprocessing table in the database, transforms it to RI's internal staging tables, then loads it into the base fact (item/location/day) as well as the week aggregate used for integrations (item/location/week).

Key Tables Affected

Table	Usage
W_RTL_INVTSF_IT_LC_DY_FTS	File Input
W_RTL_INVTSF_IT_LC_DY_FS	Staging
W_RTL_INVTSF_IT_LC_DY_F	Output (Base Fact)
W_RTL_INVTSF_IT_LC_WK_A	Output (Aggregate)

Translation Lookup Load (Legacy)

Module Name	W_DOMAIN_MEMBER_LKP_TL_PROCESS_ADHOC
Description	Processes the translatable string data in the `W_DOMAIN_MEMBER_DS_TL.dat` file and loads it into RI.
Dependencies	HIST_ZIP_FILE_LOAD_ADHOC
Business Activity	Historical Data Load

Design Overview

This process looks for the `W_DOMAIN_MEMBER_DS_TL.dat` file placed on the server by a history ZIP file upload and loads it to the target table in RI for translatable strings. When using CSV file uploads, all the translatable strings from the CSV files are automatically inserted into this table and loaded in RI without a second file being provided. However, if you are using legacy files, or you need to update records in this table directly, you can use this process to manually load string lookup records.

Key Tables Affected

Table	Usage
W_DOMAIN_MEMBER_DS_TL	Staging
W_DOMAIN_MEMBER_LKP_TL	Output

3

AI Foundation Cloud Services Standalone Processes

The primary function of standalone processes in the AI Foundation Cloud Services is move data from RI or external sources into the platform, or to move data out of the platform to send it elsewhere. These process flows differ from the Retail Insights jobs in that most processes contain only one POM job. That job contains many individual programs in it, but the execution flow is determined by parameters passed into the job. This is done by editing the job's parameters from the Batch Monitoring screen in POM:

Edit RSE_MASTER_ADHOC_JOB	✕
Enable Job	⬤ Enabled
Parameters	-pcltdgsPLKGz
External Status Update	None ▼
Skip On Error	◯ Disabled
* Threshold Run Time (Sec)	0 ⌄ ⌃
Notes	

Each letter in the string refers to a specific program or step in the execution flow, which will be covered in more detail in the sections of this chapter. When multiple parameters are used, such as when start/end dates are provided, the format of those parameters uses double-hyphens and colons as shown here:

Edit RSE_MASTER_ADHOC_JOB   ✕

Enable Job   🔵   Enabled

Parameters   -ix --from:20201012 --to:20201031 --extfrom:Y --extto:N

External Status Update   None ▾

Skip On Error   ⚪   Disabled

* Threshold Run Time (Sec)   0

Notes

This chapter includes the following programs:

- Customer Metrics - Base Calculation
- Customer Metrics - Final Calculation
- Customer Metrics - Loyalty Score
- Fake Customer Identification
- File Export Execution
- File Export Preparation
- Location Ranging
- Master Data Load - AA
- Master Data Load - AC
- Master Data Load - AE
- Master Data Load - Common
- Master Data Load - DT
- Master Data Load - IO
- Master Data Load - PMO
- Master Data Load - OO
- Master Data Load - SO
- Master Data Load - SPO
- Offer Optimization Run
- Product Location Ranging
- Sales Aggregation - Customer Segment
- Sales Aggregation - Product

- [Sales Aggregation - Product Attribute](#)
- [Sales Aggregation - Product Hierarchy](#)
- [Sales Aggregation - Weekly](#)
- [Sales Forecast Aggregation - Product Attribute (Legacy)](#)
- [Sales Forecast Aggregation - Product Hierarchy (Legacy)](#)
- [Sales Shares - Product Attribute](#)
- [Sales Transaction Load](#)

Customer Metrics - Base Calculation

Module Name	RSE_CUST_ENG_METRIC_BASE_ADHOC
Description	Calculate base values for customer engagement metrics.
Dependencies	RSE_SLS_TXN_ADHOC
Business Activity	Analytical Batch Processing

Design Overview

This process aggregates sales transaction data for use in customer engagement metric calculations. The process runs for a range of weeks, depending on which weeks of sales have had a run already performed. It will output the results to a database table for downstream consumption. The `RSE_SLS_TXN_ADHOC` job is normally a prerequisite for this, as it is used to refresh or load additional sales data.

Running this process requires parameters to specify the start and end date range for which data should be processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
RSE_SLS_TXN	Input
RSE_SLS_BASE_ATTR	Output

Customer Metrics - Final Calculation

Module Name	RSE_CUST_ENG_METRIC_CALC_ADHOC
Description	Finalize the customer engagement metrics calculation.
Dependencies	RSE_CUST_ENG_METRIC_BASE_ADHOC
Business Activity	Analytical Batch Processing

Design Overview

This process calculates customer engagement metrics based on numerous inputs, including sales transaction aggregates (for behavioral and predictive metrics) and product attributes (for attribute loyalty metrics). Currently, supported product attributes must have a group type of `BRAND`, `STYLE`, `COLOR`, `LOC_LOYALTY`, or `PRICE_EFF_LOYALTY`, as defined in `RSE_BUSINESS_OBJECT_ATTR_MD`. The `RSE_CUST_ENG_METRIC_BASE_ADHOC` job is normally a prerequisite for this, as it calculates the aggregated customer sales data.

Running this process requires parameters to specify the start and end date range, for which data should be processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
RSE_SLS_BASE_ATTR	Input
RSE_PROD_ATTR	Input
RSE_CUST_ATTR_LOY_DTL	Input
RSE_CUST_SLS_ATTR	Output

Customer Metrics - Loyalty Score

Module Name	RSE_CUST_ATTR_LOY_ADHOC
Description	Calculate customer loyalty score metrics.
Dependencies	RSE_CUST_ENG_METRIC_BASE_ADHOC
Business Activity	Analytical Batch Processing

Design Overview

This process calculates customer engagement loyalty data based on numerous inputs, including sales transactions and product attributes. Currently, supported product attributes must have a group type of `BRAND`, `STYLE`, `COLOR`, `LOC_LOYALTY`, or `PRICE_EFF_LOYALTY`, as defined in `RSE_BUSINESS_OBJECT_ATTR_MD`. The `RSE_CUST_ENG_METRIC_BASE_ADHOC` job is normally a prerequisite for this, as it calculates the aggregated customer sales data.

Running this process requires parameters to specify the start and end date range, for which data should be processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
RSE_SLS_BASE_ATTR	Input
RSE_PROD_ATTR	Input
RSE_CUST_ATTR_LOY_DTL	Output

Fake Customer Identification

Module Name	RSE_FAKE_CUST_ADHOC
Description	Identify fake customers by looking through sales transaction data, so they can be automatically excluded from some applications.
Dependencies	RSE_SLS_TXN_ADHOC
Business Activity	Sales Preprocessing

Design Overview

This process analyzes sales transaction data looking for "fake" customers, which usually represent excessive sales attributed to a single customer ID. This could be caused by store cards used at the register, corporate cards used by many people, or wholesale transactions involving large numbers of sales. These kinds of transactions can have negative effects on processes like Demand Transference because they are not representative of real customer activity. The threshold for identifying a customer as fake is set using the `RSE_CONFIG` property `FAKE_CUST_DAY_TXN_THRESHOLD`.

Running this routine requires parameters to specify the start and end date range, for which data should be re-processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
RSE_SLS_TXN	Input
RSE_FAKE_CUST	Output

File Export Execution

Module Name	RSE_POST_EXPORT_ADHOC
Description	Runs the export processes for any prepared AI Foundation export files, which includes file movement, zipping, and export to SFTP.
Dependencies	RSE_EXPORT_PREP_ADHOC

Business Activity	Outbound Integrations

Design Overview

This process moves, zips, and exports files from the AI Foundation applications based on the file export type. It accepts a single input parameter for the file frequency type, using one of `DAILY`, `WEEKLY`, `QUARTERLY`, `INTRADAY`, or `ADHOC`. This process is the second step in the data flow and assumes files have already been prepared for export using the dependent process.

File Export Preparation

Module Name	RSE_EXPORT_PREP_ADHOC
Description	Export preparation job for a specific group of AI Foundation export files.
Dependencies	None
Business Activity	Outbound Integrations

Design Overview

This process will prepare a set of export files from the AI Foundation applications based on the file export type. It accepts a single input parameter for the file frequency type, using one of `DAILY`, `WEEKLY`, `QUARTERLY`, `INTRADAY`, or `ADHOC`. This is the first step in the data flow and does not perform the file movement to SFTP; it only prepares the files of the specified type so that the `RSE_POST_EXPORT_ADHOC` process can consume them.

Location Ranging

Module Name	DT_LOC_RANGE_ADHOC
Description	Refresh Location Ranging data for Demand Transference.
Dependencies	DT_PROD_LOC_RANGE_ADHOC
Business Activity	Application Setup

Design Overview

This process calculates SKU Counts for the available ranges of products, for a given CM Group, Store Location, and Week, which may be needed during implementation of Demand Transference when using CM Groups.

Running this routine requires parameters to specify the start and end date range for weeks of data to process. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
DT_PROD_LOC_STATUS	Input
DT_CM_GRP_LOC_STATUS_AGGR	Output

Master Data Load - AA

Module Name	MBA_MASTER_ADHOC_PROCESS
Description	Run the Affinity Analysis/Market Basket Analysis master script. This is the best way to execute all the initial processing steps for the MBA application module.
Dependencies	RSE_MASTER_ADHOC_PROCESS
Business Activity	Initial Data Loads

Design Overview

This process controls the master set of batch programs for loading data into the Affinity Analysis (also known as Market Basket Analysis or MBA) application. It accepts one or more single-character parameters to control which steps in the process are executed. Multiple steps executed in sequence should be passed as one string.

Options:

- `-A` Process all steps

- `-R <Option>` Resume processing all steps, starting with the step associated with the provided option (see below options) for order

- `-e` Execute MBA ETL routines

- `-c` Execute ARM configuration load routines

- `-a` Execute ARM processes

- `-r` Execute RI ARM processes

- `-b` Execute Baseline processes

- `-?` Display this usage information

Options `-A` and `-R` will enable processing of appropriate steps. Any switch provided more than once, or after a `-A` or `-R` will toggle the switch On/Off. This will enable excluding a small number of steps from processing, without requiring specifying all other switches

Example:

`-Aa` will result in running all steps except `-a`

Master Data Load - AC

Module Name	CIS_MASTER_ADHOC_PROCESS

Description	Run the Advanced Clustering/Customer Segmentation master script. This is the best option to run all initial processing steps for the AC/CS modules. NOTE: when running through POM, if any -- options are required, use : instead of = to separate the option from the value.
Dependencies	RSE_MASTER_ADHOC_PROCESS
Business Activity	Initial Data Loads

Design Overview

This process controls the master set of batch programs for loading data into the Advanced Clustering and Customer Segmentation applications. It accepts one or more single-character parameters to control which steps in the process are executed. Multiple steps executed in sequence should be passed as one string.

Options:

- `-A` Process all steps

- `-R <Option>` Resume processing all steps, starting with the step associated with the provided option (see below options) for order

- `-a` Attribute Maintenance

- `-h` Product/Attribute Share Processing

- `-t` Loading cluster templates

- `-v` Setup a new version

- `-s` Update sales data for use by any versions

- `-m` Market Sales Aggregation load

- `-c` Update new versions with all the attribute summary information

- `-?` Display this usage information

Options `-A` and `-R` will enable processing of appropriate steps. Any switch provided more than once, or after a `-A` or `-R` will toggle the switch On/Off. This will enable excluding a small number of steps from processing, without requiring specifying all other switches.

Examples:

- `-Ah` will result in running all steps except -h

- `--from` Start date of the data processing timeframe. Must be provided in YYYYMMDD format with no spaces. For example, `--from:20170101`. Must be accompanied by the end date and optionally by the `extfrom` flag

- `--to` End date of the data processing timeframe. Must be provided in YYYYMMDD format with no spaces. For example, `--to:20170201`. Must be accompanied by the end date and optionally by the extto flag

- `--extfrom` Optional flag to indicate if the start date must be extended to the start of the week. Accepts Y or N (default). For example, `--extfrom:Y`, with no spaces

- `--extto` Optional flag to indicate if the end date must be extended to the end of the week. Accepts Y or N (default). For example, `--extto:Y`, with no spaces

Master Data Load - AE

Module Name	AE_MASTER_ADHOC_PROCESS
Description	Run the Attribute Extraction master script. This is the best way to trigger all initial processing for the AE application.
Dependencies	RSE_MASTER_ADHOC_PROCESS
Business Activity	Initial Data Loads

Design Overview

This process controls the master set of batch programs for loading data into the Attribute Extraction application. It accepts one or more single-character parameters to control which steps in the process are executed. Multiple steps executed in sequence should be passed as one string.

Options:

- `-A` Process all steps

- `-R <Option>` Resume processing all steps, starting with the step associated with the provided option (see below options) for order

- `-G` Global Lists of Strings loading

- `-C` Product Categories loading

- `-P` Product loading

- `-?` Display this usage information

Options `-A` and `-R` will enable processing of appropriate steps. Any switch provided more than once, or after a `-A` or `-R` will toggle the switch On/Off. This will enable excluding a small number of steps from processing, without requiring specifying all other switches.

Example:

`-AGP` will result in running all steps except `-G` and `-P`

Master Data Load - Common

Module Name	RSE_MASTER_ADHOC_PROCESS
Description	Run the AI Foundation Cloud Services common master script. This is the first step that should be run once data has been loaded into RI, and is ready to initialize data needed by all the other application modules. NOTE: when running through POM, if any -- options are required, use : instead of = to separate the option from the value.
Dependencies	None
Business Activity	Initial Data Loads

Design Overview

This process controls the master set of batch programs for loading data into the Retail AI Foundation Cloud Services foundation data tables. This process is generally required as the first step in loading data to any AI Foundation application. It accepts one or more single-character parameters to control which steps in the process are executed. Multiple steps executed in sequence should be passed as one string.

Options:

- `-A` Process all steps
- `-R <Option>` Resume processing all steps, starting with the step associated with the provided option (see below options) for order
- `-p` Product Hierarchy ETL
- `-c` CM Group Alternate Product hierarchy
- `-l` Location Hierarchy ETL
- `-t` Trade Area Alternate Location hierarchy
- `-d` Calendar Hierarchy ETL
- `-g` Customer Segment Hierarchy ETL
- `-s` Consumer segment data
- `-P` Product Attributes
- `-L` Location attributes
- `-K` Like Location / Product data load
- `-G` Customer Segment Attributes
- `-z` Price zone ETL
- `-h` Holiday data load
- `-i` Inventory data load
- `-C` Price and Cost data load
- `-x` Sales transaction data
- `-f` Fake customer data load
- `-k` Fake customer data identification
- `-w` Weekly Aggregate Sales data (Load or Calc)
- `-a` Aggregate Sales data processing
- `-F` Forecast Aggregate Sales data processing
- `-u` UDA load
- `-E` Export Group Setup
- `-W` Weather Driven Demand data load
- `-T` Weekly Return transactions
- `-e` Weekly Return Aggregation

- `-S` Weekly Sales Return Price Consolidation

- `-m` Customer Engagement Attribute

- `-o` Forecast Plan Load

- `-b` Bedget allocation data Load

- `-O` Order Cost data Load

- `-n` Promotion data Load

- `-D` Daily data Load

- `-?` Display this usage information

Options `-A` and `-R` will enable processing of appropriate steps. Any switch provided more than once, or after a `-A` or `-R` will toggle the switch On/Off. This will enable excluding a small number of steps from processing, without requiring specifying all other switches.

Examples:

- `-Act` will result in running all steps except `-c` and `-t`

- `-Rc -t` will result in running all steps starting with `c`, but excluding step `t`

- `--from` Start date of the data processing timeframe. Must be provided in YYYYMMDD format with no spaces. For example `--from:20170101`. Must be accompanied by the end date and optionally by the `extfrom` flag

- `--to` End date of the data processing timeframe. Must be provided in YYYYMMDD format with no spaces. For example, `--to:20170201`. Must be accompanied by the end date and optionally by the `extto` flag

- `--extfrom` Optional flag to indicate whether the start date must be extended to the start of the week. Accepts `Y` or `N` (default). For example, `--extfrom:Y`, with no spaces

- `--extto` Optional flag to indicate whether the end date must be extended to the end of the week. Accepts `Y` or `N` (default). For example, `--extto:Y`, with no spaces

Master Data Load - DT

Module Name	DT_MASTER_ADHOC_PROCESS
Description	Run the Demand Transference master script. This is the best way to run all the initial processing steps needed by the DT application module. NOTE: when running through POM, if any -- options are required, use : instead of = to separate the option from the value.
Dependencies	RSE_MASTER_ADHOC_PROCESS
Business Activity	Initial Data Loads

Design Overview

This process controls the master set of batch programs for loading data into the Demand Transference application. It accepts one or more single-character parameters to control which steps in the process are executed. Multiple steps executed in sequence should be passed as one string.

Options:

- `-A` Process all steps

- `-R <Option>` Resume processing all steps, starting with the step associated with the provided option (see below options) for order

- `-r` Load Store Sku Ranging Data

- `-l` Aggregate Location Ranging Statistics

- `-b` Calculate Baseline

- `-i` Update model intervals

- `-g` Run Group Load

- `-?` Display this usage information

Options `-A` and `-R` will enable processing of appropriate steps. Any switch provided more than once, or after a `-A` or `-R` will toggle the switch On/Off. This will enable excluding a small number of steps from processing, without requiring specifying all other switches.

Examples:

- `-Ab` will result in running all steps except `-b`

- `--from` Start date of the data processing timeframe. Must be provided in YYYYMMDD format with no spaces. For example, `--from:20170101`. Must be accompanied by the end date and optionally by the `extfrom` flag

- `--to` End date of the data processing timeframe. Must be provided in YYYYMMDD format with no spaces. For example, `--to:20170201`. Must be accompanied by the end date and optionally by the `extto` flag

- `--extfrom` Optional flag to indicate if the start date must be extended to the start of the week. Accepts `Y` or `N` (default). For example, `--extfrom:Y`, with no spaces

- `--extto` Optional flag to indicate if the end date must be extended to the end of the week. Accepts `Y` or `N` (default). For example, `--extto:Y`, with no spaces

Master Data Load - IO

Module Name	IO_MASTER_ADHOC_PROCESS
Description	Run the Inventory Optimization master script. This is the best option for running all the intial processing steps needed by the IO application module.
Dependencies	RSE_MASTER_ADHOC_PROCESS
Business Activity	Initial Data Loads

Design Overview

This process controls the master set of batch programs for loading data into the Inventory Optimization application. It accepts one or more single-character parameters to control which steps in the process are executed. Multiple steps executed in sequence should be passed as one string.

Options:

- `-A` Process all steps

- `-R <Option>` Resume processing all steps, starting with the step associated with the provided option (see below options) for order

- `-a` Replenishment Attributes at Product/Location or Group level

- `-w` Replenishment Attributes at Product/Warehouse or Group level

- `-n` Non-receiving Dates for Locations

- `-N` Non-receiving Dates for Location Types

- `-d` Non-receiving Days for Locations

- `-t` Warehouse Source Split Target

- `-s` Seasons

- `-c` Shipping Costs

- `-r` Strategy Rules

- `-?` Display this usage information

Options `-A` and `-R` will enable processing of appropriate steps. Any switch provided more than once, or after a `-A` or `-R` will toggle the switch On/Off. This will enable excluding a small number of steps from processing, without requiring specifying all other switches.

Example:

`-AbP` will result in running all steps except `-b` and `-P`

Master Data Load - PMO

Module Name	PMO_MASTER_ADHOC_PROCESS
Description	Run the Pricing and Markdown Optimization master script. This is the best way to perform all initial processing steps needed by the PMO application module.
Dependencies	RSE_MASTER_ADHOC_PROCESS
Business Activity	Initial Data Loads

Design Overview

This process controls the master set of batch programs for loading data into the Pricing and Markdown Optimization application. It accepts one or more single-character parameters to control which steps in the process are executed. Multiple steps executed in sequence should be passed as one string.

Options:

- `-A` Process all steps

- `-R <Option>` Resume processing all steps, starting with the step associated with the provided option (see below options) for order

- `-a` Activities

- `-d` Return Data Preparation

- `-c` Return Calculation

- `-h` Holiday load

- `-?` Display this usage information

Options `-A` and `-R` will enable processing of appropriate steps. Any switch provided more than once, or after a `-A` or `-R` will toggle the switch On/Off. This will enable excluding a small number of steps from processing, without requiring specifying all other switches.

Example:

`-Adh` will result in running all steps except `-d` and `-h`

Master Data Load - OO

Module Name	PRO_MASTER_ADHOC_PROCESS
Description	Run the Offer Optimization master script. This is the best option for running all the initial processing steps for the OO application module.
Dependencies	RSE_MASTER_ADHOC_PROCESS
Business Activity	Initial Data Loads

Design Overview

This process controls the master set of batch programs for loading data into the Offer Optimization application. It accepts one or more single-character parameters to control which steps in the process are executed. Multiple steps executed in sequence should be passed as one string.

Options:

- `-A` Process all steps

- `-R <Option>` Resume processing all steps, starting with the step associated with the provided option (see below options) for order

- `-b` Baseline

- `-c` Customer Segment Lifetime Value

- `-i` Inventory Aggregation

- `-f` Lifecycle Fatigue

- `-p` Promotion

- `-l` Promotion Lift

- `-C` Price-Cost

- `-e` Price Elasticity

- `-L` Price Ladder

- `-r` Sales Return

- `-s` Season

- `-P` Season Product

- `-d` Season Period

- `-m` Season Metrics

- `-E` Markdown Day of Week

- `-y` Seasonality

- `-D` Model Dates

- `-O` Country Locale

- `-F` Forecast Adjustment

- `-W` Days of Week Profile

- `-u` Properties and Rules

- `-?` Display this usage information

Options `-A` and `-R` will enable processing of appropriate steps. Any switch provided more than once, or after a `-A` or `-R` will toggle the switch On/Off. This will enable excluding a small number of steps from processing, without requiring specifying all other switches.

Example:

`-AbP` will result in running all steps except `-b` and `-P`

Master Data Load - SO

Module Name	SO_MASTER_ADHOC_PROCESS
Description	Run the Space Optimization master script. This is the best way to run all the initial steps for the SO application module.
Dependencies	RSE_MASTER_ADHOC_PROCESS
Business Activity	Initial Data Loads

Design Overview

This process controls the master set of batch programs for loading data into the Offer Optimization application. It accepts one or more single-character parameters to control which steps in the process are executed. Multiple steps executed in sequence should be passed as one string.

Options:

- `-A` Process all steps

- `-R` `<Option>` Resume processing all steps, starting with the step associated with the provided option (see below options) for order

- `-F` Assortment Finalization

- `-a` Assortment

- `-h` Placeholder Product Loading

- `-M` Product Cluster mapping

- `-C` Assortment product location forecast and price/cost

- `-f` Assortment Forecast loading

- `-r` Replenishment Parameters

- `-S` Product Stacking Height Limit

- `-p` Pog Loading

- `-b` Bay/Fixture Loading

- `-y` Display Style Loading

- `-c` Product Fixture Configuration Loading

- `-P` Perform Product Attribute maintenance

- `-m` Assortment Mapping

- `-v` Global Validation

- `-s` Assortment to POG mapping

- `-g` POG Set location creation

- `-?` Display this usage information

Options `-A` and `-R` will enable processing of appropriate steps. Any switch provided more than once, or after a `-A` or `-R` will toggle the switch On/Off. This will enable excluding a small number of steps from processing, without requiring specifying all other switches.

Example:

`-AaP` will result in running all steps except `-a` and `-P`

Master Data Load - SPO

Module Name	SPO_MASTER_ADHOC_PROCESS
Description	Run the Size Profile Optimization master script. This is the best way to run all the initial processing steps needed by the SPO application module. NOTE: when running through POM, if any -- options are required, use : instead of = to separate the option from the value.
Dependencies	RSE_MASTER_ADHOC_PROCESS
Business Activity	Initial Data Loads

Design Overview

This process controls the master set of batch programs for loading data into the Size Profile Optimization application. It accepts one or more single-character parameters to control which steps in the process are executed. Multiple steps executed in sequence should be passed as one string.

Options:

- `-A` Process all steps

- `-R` `<Option>` Resume processing all steps, starting with the step associated with the provided option (see below options) for order

- `-S` Season Data Load

- `-r` Size Range Data Load

- `-s` Size Data Load

- `-p` Product Size Data Load

- `-l` Sub-Size Range Product Location Data Load

- `-?` Display this usage information

Options `-A` and `-R` will enable processing of appropriate steps. Any switch provided more than once, or after a `-A` or `-R` will toggle the switch On/Off. This will enable excluding a small number of steps from processing, without requiring specifying all other switches.

Example:

- `-Ar` will result in running all steps except `-r`

- `--from` Start date of the data processing timeframe. Must be provided in YYYYMMDD format with no spaces. For example, `--from:20170101`. Must be accompanied by the end date and optionally by the `extfrom` flag

- `--to` End date of the data processing timeframe. Must be provided in YYYYMMDD format with no spaces. For example, `--to:20170201`. Must be accompanied by the end date and optionally by the `extto` flag

- `--extfrom` Optional flag to indicate if the start date must be extended to the start of the week. Accepts `Y` or `N` (default). For example, `--extfrom:Y`, with no spaces

- `--extto` Optional flag to indicate if the end date must be extended to the end of the week. Accepts `Y` or `N` (default). For example, `--extto:Y`, with no spaces

Offer Optimization Run

Module Name	PRO_OPT_ADHOC
Description	Runs the offer optimization process outside of the normal batch.
Dependencies	None
Business Activity	Analytical Batch Processing

Design Overview

This process triggers the offer optimization batch processing outside of the normal batch window. All of the necessary steps to calculate optimization results are included in the ad oc job and no parameters are used. The process triggers the Java libraries on the application server that are responsible for the optimization.

Product Location Ranging

Module Name	DT_PROD_LOC_RANGE_ADHOC
Description	Refresh Product Location Ranging data for Demand Transference.
Dependencies	W_RTL_IT_LC_D_JOB (in RI)
Business Activity	Application Setup

Design Overview

This process extracts the item/location ranging information from Retail Insights table `W_RTL_IT_LC_D`. This process is also performed in the DT master batch process, but it can be run on its own if you are modifying the data and need to reload it.

Running this routine requires parameters to specify the start and end date range, for which data should be re-processed from the `W_RTL_IT_LC_D` table or from AI Foundation sales tables. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
W_RTL_IT_LC_D	Input
DT_PROD_LOC_LAST_SLS	Input
DT_PROD_LOC_STATUS	Output

Sales Aggregation - Customer Segment

Module Name	RSE_WKLY_SLS_CUST_SEG_ADHOC
Description	Aggregates Sales Transaction data to Weekly Customer Segment Sales tables.
Dependencies	RSE_SLS_TXN_ADHOC
Business Activity	Initial Data Loads

Design Overview

This process aggregates sales data by customer segment for use in AI Foundation applications. The `RSE_SLS_TXN_ADHOC` job is normally a prerequisite for this, as it is used to refresh or load additional sales data.

Running this process requires parameters to specify the start and end date range, for which data should be processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
RSE_SLS_TXN	Input
RSE_CUSTSEG_CUST_XREF	Input

Table	Usage
RSE_CUSTSEG_SRC_XREF	Input
RSE_SLS_PR_LC_CS_WK	Output

Sales Aggregation - Product

Module Name	RSE_WKLY_SLS_PR_AGGR_ADHOC
Description	Calculates Product-based sales aggregate tables.
Dependencies	RSE_WKLY_SLS_ADHOC
Business Activity	Initial Data Loads

Design Overview

This process aggregates sales data by product for use in AI Foundation applications. The `RSE_WKLY_SLS_ADHOC` job is normally a prerequisite for this, as it is used to refresh or load additional sales data.

Running this process requires parameters to specify the start and end date range, for which data should be processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
RSE_SLS_PR_LC_WK	Input
RSE_SLS_PR_WK_A	Output

Sales Aggregation - Product Attribute

Module Name	RSE_WKLY_SLS_PH_ATTR_AGGR_ADHOC
Description	Calculates Product Attribute-based sales aggregate tables.
Dependencies	RSE_WKLY_SLS_ADHOC
Business Activity	Initial Data Loads

Design Overview

This process aggregates sales data by product attribute and product hierarchy levels for use in AI Foundation applications. The `RSE_WKLY_SLS_ADHOC` job is normally a prerequisite for this, as it is used to refresh or load additional sales data.

Running this process requires parameters to specify the start and end date range, for which data should be processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
RSE_SLS_PR_LC_WK	Input
RSE_PROD_ATTR	Input
RSE_SLS_PH_ATTR_LC_WK_A	Output

Sales Aggregation - Product Hierarchy

Module Name	RSE_WKLY_SLS_PH_AGGR_ADHOC
Description	Calculates Product Hierarchy-based sales aggregate tables.
Dependencies	RSE_WKLY_SLS_ADHOC
Business Activity	Initial Data Loads

Design Overview

This process aggregates sales data by product hierarchy levels for use in AI Foundation applications. The `RSE_WKLY_SLS_ADHOC` job is normally a prerequisite for this, as it is used to refresh or load additional sales data.

Running this process requires parameters to specify the start and end date range, for which data should be processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
RSE_SLS_PR_LC_WK	Input
RSE_SLS_PH_LC_WK_A	Output

Sales Aggregation - Weekly

Module Name	RSE_WKLY_SLS_ADHOC
Description	Aggregates Sales Transaction data to week level tables.
Dependencies	RSE_SLS_TXN_ADHOC

Business Activity	Initial Data Loads

Design Overview

This process aggregates sales data by product hierarchy levels for use in AI Foundation applications. The `RSE_SLS_TXN_ADHOC` job is normally a prerequisite for this, as it is used to refresh or load additional sales data.

Running this process requires parameters to specify the start and end date range, for which data should be processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
RSE_SLS_TXN	Input
RSE_SLS_PR_LC_WK	Output

Sales Forecast Aggregation - Product Attribute (Legacy)

Module Name	RSE_SLSFC_PH_ATTR_AGGR_ADHOC
Description	Calculates Product Attribute-based sales forecast aggregate tables.
Dependencies	RSE_SLSFC_PH_AGGR_ADHOC
Business Activity	Initial Data Loads

Design Overview

This process aggregates sales forecast data by product attribute and product hierarchy levels for use in AI Foundation applications. The `RSE_SLSFC_PH_AGGR_ADHOC` job is normally a prerequisite for this, as it is used to refresh or load additional sales forecast data.

Running this process requires parameters to specify the start and end date range, for which data should be processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

> **Note:**
> This is a legacy process which uses a forecast interface from RI that has been deprecated.

Sales Forecast Aggregation - Product Hierarchy (Legacy)

Module Name	RSE_SLSFC_PH_AGGR_ADHOC
Description	Calculates Product Hierarchy-based sales forecast aggregate tables.
Dependencies	None
Business Activity	Initial Data Loads

Design Overview

This process aggregates sales forecast data by product hierarchy levels for use in AI Foundation applications.

Running this process requires parameters to specify the start and end date range, for which data should be processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

> **Note:**
>
> This is a legacy process which uses a forecast interface from RI that has been deprecated.

Sales Shares - Product Attribute

Module Name	AC_PROD_ATTR_LOC_SHARE_ADHOC
Description	Calculate product attribute sales shares for use in Advanced Clustering.
Dependencies	RSE_WKLY_SLS_ADHOC
Business Activity	Initial Data Loads

Design Overview

This process aggregates sales shares by product attribute for use in the Advanced Clustering application, specifically for use in clustering by product attribute. The `RSE_WKLY_SLS_ADHOC` job is normally a prerequisite for this, as it is used to refresh or load additional sales data at week level.

You also must choose which attribute mode is applicable for AC. If it is specified as `CDT` in `RSE_CONFIG` property `PERF_CIS_APPROACH`, then this program will expect additional information for CDT-like attribute groups in `RSE_PROD_ATTR_GRP` and `RSE_PROD_ATTR_GRP_VALUE_MAP`. It will also use sales data from `RSE_SLS_PH_ATTR_LC_WK_A`. For any other configuration, these tables are not required and a more generic approach will be taken.

Running this process requires parameters to specify the start and end date range, for which data should be processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
CIS_TCRITERIA_ATTR	Input
CIS_BUS_OBJ_TCRITERIA_ATT_XREF	Input
RSE_PROD_ATTR_GRP	Input
RSE_PROD_ATTR_GRP_VALUE_MAP	Input
RSE_SLS_PH_LC_WK_A	Input
RSE_SLS_PH_ATTR_LC_WK_A	Input
CIS_PROD_ATTR_LOC_SHARE	Output

Sales Transaction Load

Module Name	RSE_SLS_TXN_ADHOC
Description	Performs bulk retrieval of Sales Transaction data.
Dependencies	W_RTL_SLS_TRX_IT_LC_DY_F_JOB (in RI)
Business Activity	Initial Data Loads

Design Overview

This process extracts sales transactions from Retail Insights for use in all AI Foundation applications. The `W_RTL_SLS_TRX_IT_LC_DY_F` table in RI is the source of this data and RI must be populated with sales before this program runs.

Running this process requires parameters to specify the start and end date range, for which data should be processed. The `-s` Parameter is for the Start date and the `-e` parameter provides the End date. Both are in format YYYYMMDD. For example:

```
-s YYYYMMDD -e YYYYMMDD -f Y
```

Key Tables Affected

Table	Usage
W_RTL_SLS_TRX_IT_LC_DY_F	Input
RSE_SLS_TXN	Output

4
Retail Insights Standalone Process Flows

Standalone processes are separated across several different flows within POM depending on the order in which you need to execute them and the dependencies to be followed. This chapter provides a tabular view of related jobs across the different process flows so you know which jobs are safe to enable or disable, depending on the files you're attempting to load into the platform. It is always a best practice to disable jobs in POM if you are not actively using them, both to reduce runtimes and to avoid loading data you did not intend to process.

Process Flows for DAT Files

The table below shows the standalone process flows for any file with a `.dat` extension. Please note that the following list of jobs should remain enabled and are usually applicable to all batch runs, so they're not included in the table.

- VARIABLE_REFRESH_JOB
- ETL_REFRESH_JOB
- ETL_BUSINESS_DATE_JOB
- W_RTL_CURR_MCAL_G_JOB
- SETUP_CTX_FILE_JOB

Table 4-1   DAT File Process Flows

Input Files (DAT)	LOAD_DIM_INITIAL_STAGE_ADHOC	LOAD_DIM_INITIAL_ADHOC
W_DOMAIN_MEMBER_DS_TL.dat	W_DOMAIN_MEMBER_DS_TL_COPY_JOB W_DOMAIN_MEMBER_DS_TL_STG_JOB	W_DOMAIN_MEMBER_LKP_TL_JOB
W_EMPLOYEE_DS.dat	W_EMPLOYEE_DS_COPY_JOB W_EMPLOYEE_DS_STG_JOB	W_EMPLOYEE_D_JOB
W_EXCH_RATE_GS.dat	W_EXCH_RATE_GS_COPY_JOB W_EXCH_RATE_GS_STG_JOB	W_EXCH_RATE_G_JOB
W_INT_ORG_ATTR_DS.dat W_INT_ORG_DS.dat W_INT_ORG_DS_TL.dat	W_INT_ORG_ATTR_DS_COPY_JOB W_INT_ORG_ATTR_DS_STG_JOB W_INT_ORG_DS_COPY_JOB W_INT_ORG_DS_STG_JOB W_INT_ORG_DS_TL_COPY_JOB W_INT_ORG_DS_TL_STG_JOB	W_INT_ORG_D_JOB W_RTL_ORG_RECLASS_TMP_JOB
W_INT_ORG_DHS.dat	W_INT_ORG_DHS_COPY_JOB W_INT_ORG_DHS_STG_JOB	W_INT_ORG_DH_JOB W_INT_ORG_DH_RTL_TMP_JOB

Table 4-1   (Cont.) DAT File Process Flows

Input Files (DAT)	LOAD_DIM_INITIAL_STAGE_AD HOC	LOAD_DIM_INITIAL_ADHOC
W_PARTY_ATTR_DS.dat	W_PARTY_ATTR_DS_COPY_JOB W_PARTY_ATTR_DS_STG_JOB	W_PARTY_ATTR_D_JOB
W_PARTY_ORG_DS.dat	W_PARTY_ORG_DS_COPY_JOB W_PARTY_ORG_DS_STG_JOB	W_PARTY_ORG_D_JOB
W_PARTY_PER_DS.dat	W_PARTY_PER_DS_COPY_JOB W_PARTY_PER_DS_STG_JOB	W_PARTY_PER_D_JOB
W_PROD_CAT_DHS.dat	W_PROD_CAT_DHS_COPY_JOB W_PROD_CAT_DHS_STG_JOB	W_PROD_CAT_DH_JOB W_PROD_CAT_DH_SC_RTL_TM P_JOB
W_PRODUCT_ATTR_DS.dat	W_PRODUCT_ATTR_DS_COPY_JO B W_PRODUCT_ATTR_DS_STG_JOB	W_PRODUCT_ATTR_D_JOB W_RTL_PRODUCT_ATTR_UDA_ D_JOB
W_PRODUCT_DS.dat W_PRODUCT_DS_TL.dat	W_PRODUCT_DS_COPY_JOB W_PRODUCT_DS_STG_JOB W_PRODUCT_DS_TL_COPY_JOB W_PRODUCT_DS_TL_STG_JOB	W_PRODUCT_D_JOB W_RTL_PROD_RECLASS_TMP_I NITIAL_JOB W_PRODUCT_D_RTL_TMP_JOB W_RTL_PROD_RECLASS_TMP_J OB
W_REASON_DS.dat	W_REASON_DS_COPY_JOB W_REASON_DS_STG_JOB	W_REASON_D_JOB
W_RTL_ALC_DETAILS_DS.d at	W_RTL_ALC_DETAILS_DS_COPY_J OB W_RTL_ALC_DETAILS_DS_STG_JO B	W_RTL_ALC_DETAILS_D_JOB
W_RTL_BUYER_DS.dat	W_RTL_BUYER_DS_COPY_JOB W_RTL_BUYER_DS_STG_JOB	W_RTL_BUYER_D_JOB
W_RTL_CHANNEL_DS.dat	W_RTL_CHANNEL_DS_COPY_JOB W_RTL_CHANNEL_DS_STG_JOB	W_RTL_CHANNEL_D_JOB
W_RTL_CO_HEAD_DS.dat	W_RTL_CO_HEAD_DS_COPY_JOB W_RTL_CO_HEAD_DS_STG_JOB	W_RTL_CO_HEAD_D_JOB
W_RTL_CO_LINE_DS.dat	W_RTL_CO_LINE_DS_COPY_JOB W_RTL_CO_LINE_DS_STG_JOB	W_RTL_CO_LINE_D_JOB
W_RTL_CO_SHIP_METHOD_ DS.dat	W_RTL_CO_SHIP_METHOD_DS_C OPY_JOB W_RTL_CO_SHIP_METHOD_DS_S TG_JOB	W_RTL_CO_SHIP_METHOD_D_ JOB
W_RTL_CO_SHIP_TYPE_DS.d at	W_RTL_CO_SHIP_TYPE_DS_COPY _JOB W_RTL_CO_SHIP_TYPE_DS_STG_J OB	W_RTL_CO_SHIP_TYPE_D_JOB
W_RTL_CODE_DS.dat	W_RTL_CODE_DS_COPY_JOB W_RTL_CODE_DS_STG_JOB	W_RTL_COMP_STORE_D_JOB

Table 4-1   (Cont.) DAT File Process Flows

Input Files (DAT)	LOAD_DIM_INITIAL_STAGE_AD HOC	LOAD_DIM_INITIAL_ADHOC
W_RTL_COUPON_DS.dat W_RTL_COUPON_DS_TL.dat	W_RTL_COUPON_DS_COPY_JOB W_RTL_COUPON_DS_STG_JOB W_RTL_COUPON_DS_TL_COPY_J OB W_RTL_COUPON_DS_TL_STG_JOB	W_RTL_COUPON_D_JOB
W_RTL_DIFF_GRP_DS.dat W_RTL_DIFF_GRP_DS_TL.da t	W_RTL_DIFF_GRP_DS_COPY_JOB W_RTL_DIFF_GRP_DS_STG_JOB W_RTL_DIFF_GRP_DS_TL_COPY_J OB W_RTL_DIFF_GRP_DS_TL_STG_JO B	W_RTL_DIFF_GRP_D_JOB
W_RTL_DIFF_RNG_DS.dat W_RTL_DIFF_RNG_DS_TL.d at	W_RTL_DIFF_RNG_DS_COPY_JOB W_RTL_DIFF_RNG_DS_STG_JOB W_RTL_DIFF_RNG_DS_TL_COPY_J OB W_RTL_DIFF_RNG_DS_TL_STG_JO B	W_RTL_DIFF_RNG_D_JOB
W_RTL_DISCOUNT_TYPE_D S.dat	W_RTL_DISCOUNT_TYPE_DS_COP Y_JOB W_RTL_DISCOUNT_TYPE_DS_STG _JOB	W_RTL_DISCOUNT_TYPE_D_JO B
W_RTL_IT_SUPPLIER_DS.dat	W_RTL_IT_SUPPLIER_DS_COPY_J OB W_RTL_IT_SUPPLIER_DS_STG_JO B	W_RTL_IT_SUPPLIER_D_JOB

Table 4-1   (Cont.) DAT File Process Flows

Input Files (DAT)	LOAD_DIM_INITIAL_STAGE_AD HOC	LOAD_DIM_INITIAL_ADHOC
W_RTL_ITEM_GRP1_DS.dat	W_RTL_ITEM_GRP1_DS_COPY_JO B	W_RTL_ITEM_GRP1_D_ITEMB RAND_JOB
	W_RTL_ITEM_GRP1_DS_STG_JOB	W_RTL_ITEM_GRP1_D_ITEMC OLOR_JOB
		W_RTL_ITEM_GRP1_D_ITEMDI FF_JOB
		W_RTL_ITEM_GRP1_D_ITEMFA B_JOB
		W_RTL_ITEM_GRP1_D_ITEMLI ST_JOB
		W_RTL_ITEM_GRP1_D_ITEML V_JOB
		W_RTL_ITEM_GRP1_D_ITEMSC EN_JOB
		W_RTL_ITEM_GRP1_D_ITEMSI ZE_JOB
		W_RTL_ITEM_GRP1_D_ITEMST YLE_JOB
		W_RTL_ITEM_GRP1_D_ITEMU DA_JOB
		W_RTL_PRODUCT_ATTR_UDA_ D_JOB
W_RTL_LOC_STOCK_CNT_D S.dat	W_RTL_LOC_STOCK_CNT_DS_CO PY_JOB	W_RTL_LOC_STOCK_CNT_D_JO B
	W_RTL_LOC_STOCK_CNT_DS_STG _JOB	
W_RTL_ORG_FIN_DS.dat	W_RTL_ORG_FIN_DS_COPY_JOB	W_RTL_ORG_FIN_D_JOB
	W_RTL_ORG_FIN_DS_STG_JOB	
W_RTL_PARTY_PER_ATTR_D S.dat	W_RTL_PARTY_PER_ATTR_DS_CO PY_JOB	W_PARTY_ATTR_D_UDA_JOB
	W_RTL_PARTY_PER_ATTR_DS_ST G_JOB	
W_RTL_PHASE_DS.dat	W_RTL_PHASE_DS_COPY_JOB	W_RTL_PHASE_D_JOB
	W_RTL_PHASE_DS_STG_JOB	
W_RTL_PO_DETAILS_DS.dat	W_RTL_PO_DETAILS_DS_COPY_J OB	W_RTL_PO_DETAILS_D_JOB
	W_RTL_PO_DETAILS_DS_STG_JOB	
W_RTL_PRICE_CLR_IT_LC_D S.dat	W_RTL_PRICE_CLR_IT_LC_DS_CO PY_JOB	W_RTL_PRICE_CLR_IT_LC_D_J OB
	W_RTL_PRICE_CLR_IT_LC_DS_ST G_JOB	
W_RTL_PROD_HIER_ATTR_L KP_DHS.dat	W_RTL_PROD_HIER_ATTR_LKP_D HS_COPY_JOB	W_RTL_PROD_HIER_ATTR_LK P_DH_JOB
	W_RTL_PROD_HIER_ATTR_LKP_D HS_STG_JOB	W_RTL_PROD_HIER_ATTR_LK P_DH_IM_JOB

Table 4-1   (Cont.) DAT File Process Flows

Input Files (DAT)	LOAD_DIM_INITIAL_STAGE_AD HOC	LOAD_DIM_INITIAL_ADHOC
W_RTL_PRODUCT_BRAND_ DS.dat	W_RTL_PRODUCT_BRAND_DS_CO PY_JOB	W_RTL_PRODUCT_BRAND_D_J OB
W_RTL_PRODUCT_BRAND_ DS_TL.dat	W_RTL_PRODUCT_BRAND_DS_ST G_JOB	
	W_RTL_PRODUCT_BRAND_DS_TL _COPY_JOB	
	W_RTL_PRODUCT_BRAND_DS_TL _STG_JOB	
W_RTL_PROMO_DS.dat	W_RTL_PROMO_DS_COPY_JOB	W_RTL_PROMO_D_TL_JOB
W_RTL_PROMO_DS_TL.dat	W_RTL_PROMO_DS_STG_JOB	W_PROMO_D_RTL_TMP_JOB
W_RTL_PROMO_CE_DS.dat	W_RTL_PROMO_CE_DS_COPY_JO B	W_RTL_PROMO_D_TL_JOB
		W_PROMO_D_RTL_TMP_JOB
	W_RTL_PROMO_CE_DS_STG_JOB	
W_RTL_PROMO_EXT_DS.dat	W_RTL_PROMO_EXT_DS_COPY_J OB	W_RTL_PROMO_D_TL_JOB
		W_PROMO_D_RTL_TMP_JOB
	W_RTL_PROMO_EXT_DS_STG_JO B	
W_RTL_SEASON_DS.dat	W_RTL_SEASON_DS_COPY_JOB	W_RTL_SEASON_D_JOB
	W_RTL_SEASON_DS_STG_JOB	
W_RTL_SEASON_PHASE_IT_ DS.dat	W_RTL_SEASON_PHASE_IT_DS_C OPY_JOB	W_RTL_SEASON_PHASE_IT_D_ JOB
	W_RTL_SEASON_PHASE_IT_DS_S TG_JOB	
W_RTL_TNDR_TYPE_DS.dat	W_RTL_TNDR_TYPE_DS_COPY_JO B	W_RTL_TNDR_TYPE_D_JOB
	W_RTL_TNDR_TYPE_DS_STG_JOB	
W_STATUS_DS.dat	W_STATUS_DS_COPY_JOB	W_STATUS_D_JOB
	W_STATUS_DS_STG_JOB	

Process Flows for CSV Files

The table below shows the standalone process flows for any file with a `.csv` extension. CSV files follow a different load path as they are simplified interfaces that are transformed from one input file to many output tables in the data model. Please note that the following list of jobs should remain enabled and are usually applicable to all batch runs, so they're not included in the table.

- VARIABLE_REFRESH_JOB

- ETL_REFRESH_JOB

- ETL_BUSINESS_DATE_JOB

- W_RTL_CURR_MCAL_G_JOB

- SETUP_CTX_FILE_JOB

- SI_W_DOMAIN_MEMBER_DS_TL_TRUNC_JOB

The `W_DOMAIN_MEMBER_DS_TL` table works differently from other loaders, as multiple jobs are inserting into the same staging area for different sets of records. The job above is needed at the start of a process flow to truncate the `W_DOMAIN_MEMBER_DS_TL` table before inserting new records in all later steps in `LOAD_EXT_DIM_INITIAL_SI_ADHOC`. If you are loading files one at a time, make sure you do not truncate `W_DOMAIN_MEMBER_DS_TL` excessively. It is only needed at the beginning of a new set of file loads or when starting over after an initial load was done.

Another important note is that you will want to load files in a certain order or together as sets, depending on the data you have available. If possible, you should load all your files.

Table 4-2   CSV File Process Flows

Input Files (CSV)	LOAD_DIM_INITIAL_CSV_ADHOC	LOAD_EXT_DIM_INITIAL_SI_ADHOC	LOAD_DIM_INITIAL_ADHOC
ATTR.csv PROD_ATTR .csv	COPY_SI_ATTR_JOB	STAGING_SI_W_RTL_PRODUCT_ATTR_DS_JOB	W_RTL_PRODUCT_ATTR_D_JOB
	STG_SI_ATTR_JOB	SI_W_RTL_PRODUCT_ATTR_DS_TL_JOB	W_RTL_ITEM_GRP1_D_ITEMBRAND_JOB
	COPY_SI_PROD_ATTR_JOB	SI_W_RTL_ITEM_GRP1_DS_JOB	W_RTL_ITEM_GRP1_D_ITEMCOLOR_JOB
	STG_SI_PROD_ATTR_JOB	STAGING_SI_W_RTL_PRODATTR_ITEM_GRP1_DS_JOB	W_RTL_ITEM_GRP1_D_ITEMDIFF_JOB
		SI_ATTR_W_DOMAIN_MEMBER_DS_TL_JOB	W_RTL_ITEM_GRP1_D_ITEMFAB_JOB
		SI_PROD_ATTR_W_DOMAIN_MEMBER_DS_TL_JOB	W_RTL_ITEM_GRP1_D_ITEMLIST_JOB
		SI_W_RTL_PRODUCT_COLOR_DS_JOB	W_RTL_ITEM_GRP1_D_ITEMLV_JOB
			W_RTL_ITEM_GRP1_D_ITEMSCEN_JOB
			W_RTL_ITEM_GRP1_D_ITEMSIZE_JOB
			W_RTL_ITEM_GRP1_D_ITEMSTYLE_JOB
			W_RTL_ITEM_GRP1_D_ITEMUDA_JOB
			W_DOMAIN_MEMBER_LKP_TL_JOB
			W_RTL_PRODUCT_ATTR_UDA_D_JOB
CODES.csv	COPY_SI_CODES_JOB	STAGING_SI_W_RTL_CODE_DS_JOB	W_RTL_CODE_D_JOB
	STG_SI_CODES_JOB		
DIFF_GROUP.csv	COPY_SI_DIFF_GROUP_JOB	SI_W_RTL_DIFF_GRP_DS_JOB	W_RTL_DIFF_GRP_D_JOB
	STG_SI_DIFF_GROUP_JOB	SI_W_RTL_DIFF_GRP_DS_TL_JOB	

Table 4-2   (Cont.) CSV File Process Flows

Input Files (CSV)	LOAD_DIM_INITIAL_CSV_ADHOC	LOAD_EXT_DIM_INITIAL_SI_ADHOC	LOAD_DIM_INITIAL_ADHOC
EMPLOYEE.csv	COPY_SI_EMPLOYEE_JOB STG_SI_EMPLOYEE_JOB	SI_W_EMPLOYEE_DS_JOB	W_EMPLOYEE_D_JOB
EXCH_RATE.csv	COPY_SI_EXCH_RATE_JOB STG_SI_EXCH_RATE_JOB	SI_W_EXCH_RATE_GS_JOB	W_EXCH_RATE_G_JOB
ORGANIZATION.csv	COPY_SI_ORGANIZATION_JOB STG_SI_ORGANIZATION_JOB	SI_W_INT_ORG_DHS_JOB SI_W_INT_ORG_ATTR_DS_JOB SI_W_INT_ORG_DS_JOB SI_W_INT_ORG_DS_TL_JOB SI_W_RTL_CHANNEL_DS_JOB SI_ORG_W_DOMAIN_MEMBER_DS_TL_JOB	W_INT_ORG_DH_JOB W_INT_ORG_D_JOB W_INT_ORG_DH_RTL_TMP_JOB W_RTL_ORG_RECLASS_TMP_JOB W_RTL_CHANNEL_D_JOB W_DOMAIN_MEMBER_LKP_TL_JOB
PROD_LOC_ATTR.csv	COPY_SI_PROD_LOC_ATTR_JOB STG_SI_PROD_LOC_ATTR_JOB	SI_W_RTL_IT_LC_DS_JOB	W_RTL_IT_LC_D_JOB
PROD_LOC_REPL.csv	COPY_SI_PROD_LOC_REPL_JOB STG_SI_PROD_LOC_REPL_JOB	STAGING_SI_W_INVENTORY_PRODUCT_ATTR_DS_JOB	W_INVENTORY_PRODUCT_ATTR_D_JOB
PROD_PACK.csv	COPY_SI_PROD_PACK_JOB STG_SI_PROD_PACK_JOB	STAGING_SI_W_RTL_ITEM_GRP2_DS_JOB	W_RTL_ITEM_GRP2_D_JOB
PROD_SEASON.csv SEASON.csv	COPY_SI_SEASON_JOB STG_SI_SEASON_JOB COPY_SI_PROD_SEASON_JOB STG_SI_PROD_SEASON_JOB	STAGING_SI_W_RTL_PHASE_DS_JOB STAGING_SI_W_RTL_SEASON_DS_JOB STAGING_SI_W_RTL_SEASON_PHASE_IT_DS_JOB SI_SEASON_W_DOMAIN_MEMBER_DS_TL_JOB	W_RTL_SEASON_D_JOB W_RTL_PHASE_D_JOB W_RTL_SEASON_PHASE_IT_D_JOB W_DOMAIN_MEMBER_LKP_TL_JOB

Table 4-2   (Cont.) CSV File Process Flows

Input Files (CSV)	LOAD_DIM_INITIAL_CSV_ADHOC	LOAD_EXT_DIM_INITIAL_SI_ADHOC	LOAD_DIM_INITIAL_ADHOC
PRODUCT.csv	COPY_SI_PRODUCT_JOB STG_SI_PRODUCT_JOB	SI_W_PROD_CAT_DHS_JOB SI_W_PRODUCT_ATTR_DS_JOB SI_W_PRODUCT_DS_JOB SI_W_PRODUCT_DS_TL_JOB SI_W_RTL_IT_SUPPLIER_DS_JOB SI_W_RTL_PRODUCT_ATTR_IMG_DS_JOB SI_W_RTL_PRODUCT_BRAND_DS_JOB SI_W_RTL_PRODUCT_BRAND_DS_TL_JOB SI_PROD_W_DOMAIN_MEMBER_DS_TL_JOB	W_PROD_CAT_DH_JOB W_RTL_PROD_HIER_ATTR_LKP_DH_JOB W_PROD_CAT_DH_SC_RTL_TMP_JOB W_RTL_PROD_HIER_ATTR_LKP_DH_IM_JOB W_PRODUCT_D_JOB W_PRODUCT_ATTR_D_JOB W_RTL_PROD_RECLASS_TMP_INITIAL_JOB W_PRODUCT_D_RTL_TMP_JOB W_RTL_PROD_RECLASS_TMP_JOB W_RTL_PRODUCT_BRAND_D_JOB W_RTL_PRODUCT_ATTR_UDA_D_JOB
PROMOTION.csv	COPY_SI_PROMO_EXT_JOB STG_SI_PROMO_EXT_JOB	SI_W_RTL_PROMO_EXT_DS_JOB	W_RTL_PROMO_D_TL_JOB W_PROMO_D_RTL_TMP_JOB
STORE_COMP.csv	COPY_SI_STORE_COMP_JOB STG_SI_STORE_COMP_JOB	STAGING_SI_W_RTL_LOC_COMP_MTX_DS_JOB	W_RTL_LOC_COMP_MTX_D_JOB
SUPPLIER.csv	COPY_SI_SUPPLIER_JOB STG_SI_SUPPLIER_JOB	SI_W_PARTY_ORG_DS_JOB STAGING_SI_W_PARTY_ATTR_DS_JOB	W_PARTY_ATTR_D_JOB W_PARTY_ORG_D_JOB

5

Data Validation Framework

The foundation file interfaces (such as product and organization hierarchies) have a set of validations and error checking jobs that execute with them to ensure the data is accurate, complete, and follows all basic requirements for RAP application usage. Review the contents of this chapter to understand what validations exist and how to reconfigure them per your implementation needs.

Architecture Overview

The validation framework consists of POM batch jobs that execute the validations, and database tables that control the types of validation rules and what happens when the rule is triggered. Some validation rules may cause the POM job to fail, which means the data has a critical issue that needs to be corrected before the batch process can continue. Other rules will simply write warnings to the database but allow the batch to proceed. In both cases, there are tables that can be queried to check the validation results and determine what actions need to be taken.

The table below summarizes the POM jobs that execute the validations:

Table 5-1   Data Validation POM Jobs

Job Name	Summary
DIM_ORG_VALIDATOR_JOB	Executes validations on the Organization Hierarchy data
DIM_PROD_VALIDATOR_JOB	Executes validations on the Product Hierarchy and Product dimension data
DIM_CALENDAR_VALIDATOR_JOB	Executes validations on the Calendar Hierarchy staging data
DIM_CALENDAR_LOAD_VALIDATOR_JOB	Executes validations on the Calendar Period data after the load has been run

The jobs are included both in the nightly batch process flow and in separate ad hoc processes that can be executed as part of your historical data loads.

The configuration table for the validation rules is called `C_DIM_RULE_LIST`. You can access this table from the Control & Tactical Center's Manage System Configurations screen. This table allows you to edit the following fields:

- Set the error message resulting from a validation rule (`ERROR_DESC`)

- Set whether the POM job should have a hard failure or only capture a warning message (`ERROR_TYPE`) with a value of `F` or `W`

- Set whether it is turned on or off (`ON_IND`) with a value of `Y` or `N`

The other important field in this table is the `BAD_TBL_NAME`, which tells you where the results of the validations will be written in the case of any errors or warnings. If a failure or warning

does occur, you can directly query the database table listed in `BAD_TBL_NAME` using Data Visualizer or APEX.

Any time you execute one or more of the validation jobs, there is also a database view that summarizes the results from the job executions. This view is `RI_DIM_VALIDATION_V` and can also be queried from DV as needed. An example of the data in this view is shown below:

	ERROR_ID	BUSINESS_DT	INSERT_DT	ERROR_DESC	ERROR_TYPE	ON_IND	BAD_TBL_NAME
1	PROD_R6	06-MAY-22	16-JUN-22	NULL VALUE COUNT	W	Y	BAD_PROD_ITEM_ATTR_ROW_CNT
2	PROD_R3	06-MAY-22	16-JUN-22	PROD_HIER_NO_CHILD_LEVEL	W	Y	BAD_PROD_PROD_CAT_DHS

Using a combination of the data in `RI_DIM_VALIDATION_V` and the specified `BAD_TBL_NAME` table data, you will be able to identify the issues and take corrective action on the source data. In the case of job failures, you will need to reload the data file to proceed. It is also possible to skip the failed validation job in POM, but this should only be done if you have carefully reviewed the validation results and are confident the data will not cause any problems in your target applications.

Resolving Validation Issues

The validation rules scan your input data for a variety of common problems that may result in failures or inconsistencies in downstream applications such as AI Foundation or Planning modules. The table below describes what the rules are checking for and how to resolve the issues.

Table 5-2   Validation Rule Details

Rule ID	Explanation	Resolution
CAL_R1	The `W_MCAL_PERIOD_D` table does not contain any data after loading a calendar file. Your calendar file may have format or data issues that require correction, such as an incorrect value for `MCAL_CAL_ID` or missing dates that prevent it from loading properly.	Reload a corrected data file after reviewing the contents. All start/end date fields must be populated and all other fields should exactly match the file requirements as documented.
CAL_R2	The start and end dates for the fiscal periods are overlapping which will result in an invalid calendar.	Create and load a new calendar file where the period start and end dates are exactly aligned and don't overlap or have gaps.
CAL_R3	The `START_DT` parameter set on `C_ODI_PARAM_VW` is not less than or equal to your first calendar period start date. This may result in missing calendar data.	Update the `START_DT` parameter from the Control Center to be earlier than your fiscal calendar start date.

Table 5-2   (Cont.) Validation Rule Details

Rule ID	Explanation	Resolution
CAL_R4	Your calendar file does not contain at least 2 years prior to the current system date. Many applications on the platform require at least 2 years before and after the current calendar year (5 years total).	Load a new calendar file having at least 2 years of fiscal periods prior to the current year.
PROD_R1	Many-to-many relationships exist in your product hierarchy, which is not allowed. This is generally due to the same child ID appearing below multiple parent IDs.	Review all hierarchy levels for instances of the same ID appearing under multiple parents (such as a department belonging to two different divisions or groups) and modify the data to remove the multi-parent issues.
PROD_R2	The same product hierarchy node has multiple descriptions on different rows of the input file.	Modify your product hierarchy file such that any given hierarchy ID has the same description on all rows.
PROD_R3	A node of the product hierarchy has no children under it. This could be due to a reclass that didn't delete the old nodes, or when a new node is added but no items were created yet.	If possible, remove all cases of nodes having no children (for example, if all items are reclassed out of a subclass, delete the old subclass). Some AI Foundation functionality will fail if you attempt to run it on empty nodes.
PROD_R4	Your product hierarchy levels use alphanumeric characters for the level IDs. This is not allowed if you are implementing Retail Insights; all levels must be numbers.	If you are implementing RI, you must alter your hierarchy to only use numbers for every level above item. Other characters are not allowed.
PROD_R5	You are attempting to delete an item while also sending data for that item in other files on the same batch run. You cannot delete an item if it is still actively sending data on other input interfaces.	Re-send the deleted item file, removing any items that are still active or posting new data to RI. If the item should be deleted, then re-send the other files having that item's data to remove the item from all other files.
PROD_R6	Null values are present on product attribute columns that are critical to the operation of multiple RAP applications. This includes the item level, tran level, diff aggregate, and item parent levels.	Fill in the null values on the specified columns with non-null values wherever possible and re-send the product file. If you are okay with the null values and understand the impact then this may be ignored.
ORG_R1	Many-to-many relationships exist in your organization hierarchy, which is not allowed. This is generally due to the same child ID appearing below multiple parent IDs.	Review all hierarchy levels for instances of the same ID appearing under multiple parents (such as a district belonging to two different regions or areas) and modify the data to remove the multi-parent issues.

Table 5-2   (Cont.) Validation Rule Details

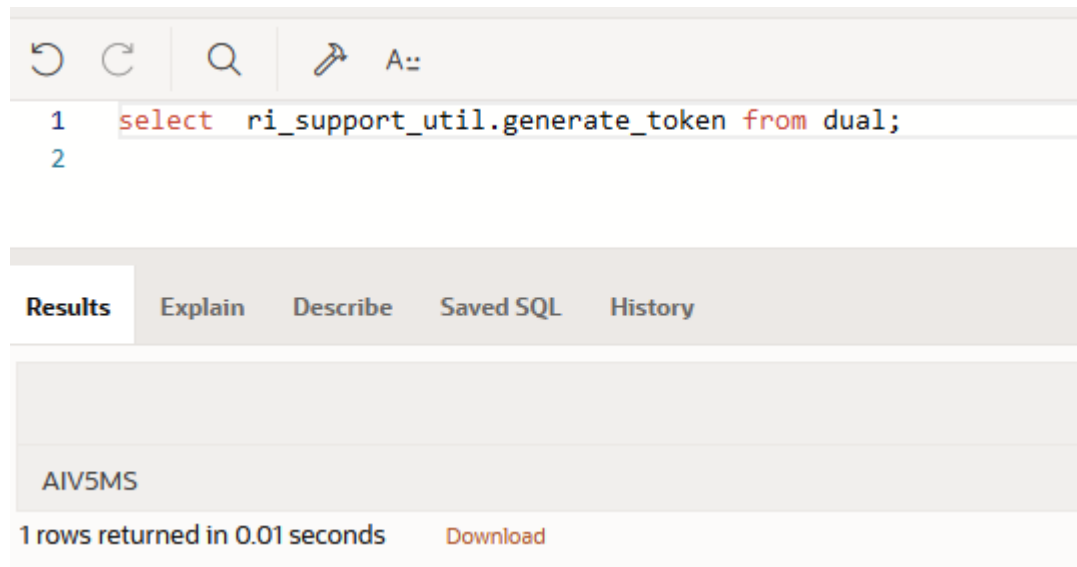| Rule ID	Explanation	Resolution
ORG_R2	The same organization hierarchy node has multiple descriptions on different rows of the input file.	Modify your organization hierarchy file such that any given hierarchy ID has the same description on all rows.
ORG_R3	A node of the organization hierarchy has no children under it. This could be due to a reclass that didn't delete the old nodes, or when a new node is added but no stores were created yet.	If possible, remove all cases of nodes having no children (for example, if all stores are reclassed out of a district, delete the old district). Some AI Foundation functionality will fail if you attempt to run it on empty nodes.
ORG_R4	Your organization hierarchy levels use alphanumeric characters for the level IDs. This is not allowed if you are implementing Retail Insights; all levels must be numbers.	If you are implementing RI, you must alter your hierarchy to only use numbers for every level of the organization hierarchy. Other characters are not allowed.

6

Support Utilities

Some support utilities will be exposed for implementers directly in APEX, allowing you to run functions such as database cleanup without Oracle involvement. These utilities may also be used by Oracle Support when responding to Service Requests on your RAP environments.

Data Cleanup Utility

Because foundation data is always loaded first through the Retail Insights data warehouse, implementers often need to erase data from the RI tables in preparation for a new load. Database functions have been exposed to APEX to allow targeted deletion of data by table name. The deletion requires a two-step process to prevent accidental execution of the command. The first step is to request an authentication token for the utility, which will return a string value. You must then pass the token into the subsequent call to the utility to execute the data cleanup step. Executing the utility with an expired or invalid token will return an error message.

The command to run in APEX to generate tokens is:

```
select  ri_support_util.generate_token from dual
```


Take the return value from this query and copy it into the `TOKEN_VALUE` variable in the PL/SQL block below. Specify the schema name and table name to be truncated, then run the PL/SQL block.

```
DECLARE
  TOKEN_VALUE VARCHAR2(200);
  SCHEMANAME VARCHAR2(200);
```

```
    TABLENAME VARCHAR2(200);
BEGIN
    TOKEN_VALUE := 'ABCDEF';
    SCHEMANAME := 'RADM01';
    TABLENAME := 'W_RTL_SLS_TRX_IT_LC_DY_FS';
    RI_SUPPORT_UTIL.CLEAR_SELECTED_RI_TABLES(
        TOKEN_VALUE => TOKEN_VALUE,
        SCHEMANAME => SCHEMANAME,
        TABLENAME => TABLENAME
    );
END;
```

If the process is successful, you will see that the PL/SQL block was successfully executed with no further message or results. If the process encounters any error, it will display the error details in the results panel in APEX. For example, if the token used is not valid it will show the following error:

```
ORA-20001: Error occurred in RI_SUPPORT_UTIL.validate_token - Error validating
token
ORA-06512: at "RADM01.RI_LOG_UTIL", line 272
ORA-06512: at "RADM01.RI_SUPPORT_UTIL", line 86
ORA-20001: Error occurred in RI_SUPPORT_UTIL.validate_token - Token Invalid.
Please try again
ORA-06512: at "RADM01.RI_LOG_UTIL", line 272
ORA-06512: at "RADM01.RI_SUPPORT_UTIL", line 81
ORA-06512: at "RADM01.RI_SUPPORT_UTIL", line 142
```

To quickly clean the entire database schema instead of individual tables, you may instead call the following command. This command will erase all customer data except for the calendar, system configuration tables, and seed data records. Use this command if you need to reset the environment in preparation for a new dataload using a different dataset:

```
DECLARE
v_token VARCHAR2(200);
SCHEMANAME VARCHAR2(200);

BEGIN
    v_token := ri_support_util.generate_token;
    SCHEMANAME := 'RADM01';
    RI_SUPPORT_UTIL.CLEAR_SELECTED_RI_TABLES(
        TOKEN_VALUE => v_token,
        SCHEMANAME => SCHEMANAME
    );
END;
```

Calendar removal is provided as a separate function, because you cannot remove calendar information without also erasing all partitions (which are specific to your currently loaded calendar). The function name is CLEAR_RI_MCAL_TABLES and can be called the same way as the schema clear script above, passing in the token and schema name as the inputs. Before you perform any calendar cleanup, review the following:

- Partition removal is based on the current partition configuration in `C_MODULE_ARTIFACT`; it will not modify tables that are not enabled for partitioning. Ensure the configuration table reflects your current cleanup needs.

- Because calendar cleanup includes partition removal, you cannot use the system for a new data load without first re-partitioning the system. Refer to the *RAP Implementation Guide* for the steps to reload the calendar and partition the database.