Oracle® Retail Analytics Platform

Implementation Guide

Release 22.1.201.0
F56032-04
July 2022

ORACLE

Oracle Retail Analytics Platform Implementation Guide, Release 22.1.201.0

F56032-04

Contents

6 Implementation Tools

List of Figures

List of Tables

Send Us Your Comments

Oracle Retail Analytics Platform Implementation Guide, Release 21.0.000

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 Implementation Guide provides critical information about the processing and operating details of the Analytics Platform, including the following:

- System configuration settings
- Technical architecture
- Functional integration dataflow across the enterprise
- Batch processing

Audience

This guide is for:

- Systems administration and operations personnel
- System analysts
- Integrators and implementers
- Business analysts who need information about Analytics Platform processes and interfaces

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

Oracle Retail Analytics Platform Cloud Service is 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

Overview

The Oracle Retail Analytics Platform is the common and extensible delivery platform for analytics and planning solutions.

The platform supports Oracle Retail applications across each of major analytical categories, including:

- Descriptive and diagnostic with merchandise, customer, and consumer insights.
- Predictive with demand forecasting, customer, and location clustering.
- Prescriptive with assortment, pricing, and inventory optimization.

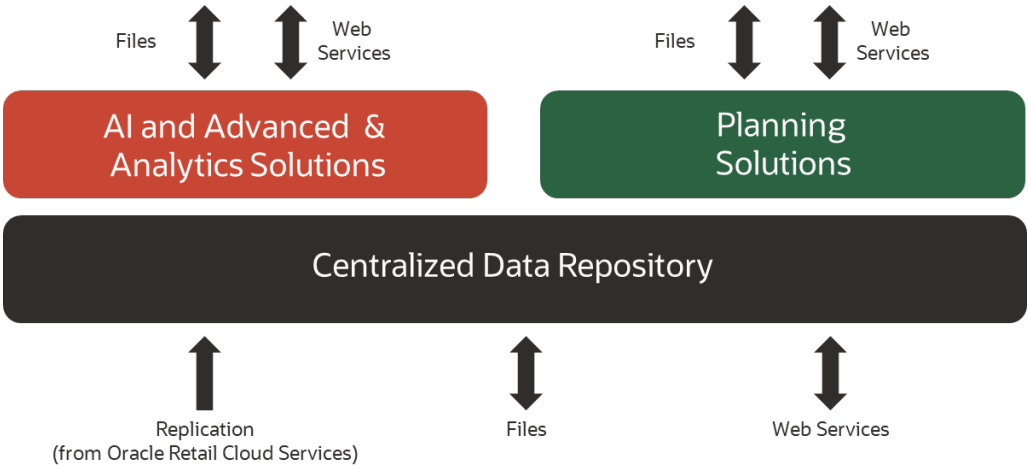
The platform also supports Oracle Retail merchandise and inventory planning solutions. These solutions support business responsiveness through a highly interactive user experience and drive the best outcomes with the application of advanced analytics and artificial intelligence (AI).

As a common platform, it provides a centralized data repository, lean integration APIs, and an efficient portfolio of delivery technologies. The data repository reflects a comprehensive data model of retail planning, operations, and execution processes. The integration APIs support right-time interactions: a lean set of bulk, on-demand, and near real-time mechanisms. The delivery technologies represent a portfolio of connected tools to build and extend composite solutions using fit-for-purpose analytical, application, and integration tools.

Architecture

The architecture of the Oracle Retail Analytics Platform provides a centralized repository and integration path for all common foundational and analytical data. The centralized repository ensures that all solutions reference consistent data definitions across transformations and aggregations. The centralized integrations simplify implementation and operational support. This centralization can also be complemented by data and integrations for customer-specific extensions to any analytics or planning solution implemented on the platform. Coordination of analytical processes and data movement within the platform is managed by Oracle Retail Process Orchestration & Monitoring using a common schedule.

The diagram below depicts the high-level platform architecture.



Getting Started

Each implementation of the Retail Analytics Platform involves one or more modules across Insights, Science, and Planning. It often includes multiple years of historical data sourced from multiple Oracle and non-Oracle systems, some of which will also need to be integrated with the platform on an ongoing basis. For many of the modules, you will also want data from the platform to be sent to other downstream applications and processes. For these reasons, every implementation is unique to your business requirements and requires careful planning and a deep understanding of all platform components.

Regardless of the modules being implemented, the outline in the table below can be followed and adapted for your project, and later chapters of the document will elaborate on many of these topics. More detailed checklists and project planning tools for some modules are also available in My Oracle Support.

Table 1-1 Implementation Outline

Project Phase	Activity	References
Pre-Implementation	Train team members on platform tools, including Retail Home, POM, APEX, Object Store, and Oracle Analytics Server (OAS).	Read the section on Implementation Tools
	Plan for the types and volumes of historical data you will need to create, based on platform and application-specific data needs.	Read the section on Data Requirements
	Plan for all inbound data sources that must provide history and ongoing data to the platform, such as external merchandising systems.	Read the section on Loading Data from Files
	Understand how data is manipulated and transformed to meet the needs of the platform modules, as this can influence your data conversion efforts.	Read the chapter on Transformations

Table 1-1 (Cont.) Implementation Outline

Project Phase	Activity	References
Environment Provisioning	Complete all activities captured in the Service Administrator Action List before starting the implementation.	Oracle Retail Analytics Platform Service Administrator Action List
	Perform the initial system setup and confirm all configurations with the customer and implementation teams.	Complete the sections on Platform Configurations and Application Configurations
Data Conversion	Extract historical data files based on the needs of your chosen RAP modules.	Complete the section on Data Requirements
	Understand how to interface with RAP to upload and move data into the platform and through to all implemented modules.	Complete the section on Preparing to Load Data
Data Loading	Initialize your system and business calendars and validate that the environment is ready to run batch processes.	Complete the section on Calendar and Partition Setup
	Integrate all data files and for history loads and initial seeding of positional data.	Complete the section on Loading Data from Files
	Ensure all implemented modules have received the necessary historical data for the entire history window.	Complete the sections on Sending Data to Science and Sending Data to Planning
Batch Processing	Enable daily and weekly batch processing and establish all ongoing integrations with Oracle and non-Oracle systems.	Complete the chapter on Batch Orchestration
Cutovers	Perform mock cutovers between Pre-Production and Production environments, where data and batches are tested in Production using final batch flows.	Environment cutovers are scheduled through My Oracle Support
	Plan for final cutover to production several weeks before the go-live date and establish an outage window based on the duration of the mock cutover.	Environment cutovers are scheduled through My Oracle Support

Based on all of the topics listed above, here is an example of some major milestones and key activities that might be followed for a project that includes Merchandise Financial Planning:

1. Oracle sends you a welcome mail having the required credentials to access your applications on the Analytics Platform - RI, RSP, MFP, POM, RH, DV, Apex, Innovation workbench.
2. Verify that the modules purchased by the customer are enabled in Retail Home during deployment. Review the steps in the *RAP Implementation Guide* on managing customer modules.
3. Verify that you can access POM and that the batch schedules for your subscribed applications are available.

4. Prepare the scripts to access object storage and test your connection to the File Transfer Services (FTS).
5. Apply initial configurations to all your applications per the documentation for each solution.
6. Upload the files required by RAP for foundation and historical data to object storage.
7. Run the first set of ad hoc jobs to load data into RI's interfaces, following the *RAP Implementation Guide* and *RAP Operations Guides* as needed.
8. Run the next set of ad hoc jobs to publish data to Science and Planning (PDS), following the *RAP Implementation Guide* and *RAP Operations Guides* as needed.
9. Repeat the ad hoc data load process iteratively until all history is loaded, and perform any needed seeding steps per the *RAP Implementation Guide* to establish full snapshots of positional data.
10. Run the PDS domain build activity to build your MFP domain.
11. Create user groups in MFP and configure access in OCI IAM for business users.
12. Configure forecast settings in RSP for generating MFP forecasts and validate forecast execution works as expected.
13. Upload files to object storage for your complete nightly batch runs and initiate the full batches for RAP.

2

Setup and Configuration

The Setup and Configuration chapter provides parameters and steps for setting up a new Retail Analytics Platform cloud environment. While the platform comprises many application modules (some of which you may not use), there are certain common processes and settings that are shared across all of them. It is critical to check and update these core settings before moving on to later implementation steps, as they will define many system-wide behaviors that could be difficult to change once you've started loading data into the platform.

Configuration Overview

A high-level outline of the setup process is provided below to describe the activities to be performed in this chapter.

Table 2-1 RAP Configuration Overview

Activity	Description
Learn the configuration tools	The Retail Analytics Platform has many tools available to support an implementation, such as Retail Home, POM, and APEX. Knowing how to use these tools is an important first step in the process. Review the Implementation Tools chapter for details.
Verify Object Storage connectivity	Generate access tokens for interacting with Object Storage and test the connection, as it is required for all file movement into and out of the Oracle cloud. Review the Implementation Tools chapter for details.
Configure the system calendar	Update the parameters that define the type and characteristics of your business calendar, such as the start and end dates RI will use to define calendar generation.
Configure the system languages	Update the master list of supported languages that need to be present in addition to your primary language, such as the need for seeing data in both English and French.
Configure history retention policies	Certain data tables in Retail Insights that are leveraged by other applications on the platform have a history retention period after which some data may be erased.
Configure application-specific settings	All applications in the Retail Analytics Platform have their own settings which must be reviewed before starting an implementation of those modules.

Platform Configurations

This section provides a list of initial setup and configuration steps to be taken as soon as you are ready to start a new implementation of the Retail Analytics Platform and have the cloud environments provisioned and generally available.

Several configuration tables in the RAP database should be reviewed before processing any data. A list of these tables is below, along with an explanation of their primary uses. The way

to apply changes to these tables is through the Control & Tactical Center, as described in the section on Control & Tactical Center.

Table 2-2 Platform Configuration Tables

Table	Usage
C_ODI_PARAM (C_ODI_PARAM_VW)	Table used to configure all Oracle Data Integrator (ODI) batch programs as well as many Retail Insights and AI Foundation load properties. `C_ODI_PARAM_VW` is the name of the table shown in Control Center.
W_LANGUAGES_G	Table used to define all languages that need to be supported in the database for translatable data (primarily for Retail Insights and Science Platform).
C_MODULE_ARTIFACT	Table used for database table partitioning setup. Defines which functional areas within RI will be populated (and thus require partitioning).
C_MODULE_EXACT_TABLE	Table used to configure partition strategies for certain tables in RI, including the Plan fact used for loading plans and budgets to RI/RSP.
C_HIST_LOAD_STATUS	Table used to configure historical data loads, configure certain ad hoc batch processes, and monitor the results of those jobs after each run.
C_SOURCE_CDC	Table used to configure and monitor both historical and ongoing integration to Planning applications through the Retail Insights data warehouse.

C_ODI_PARAM Initialization

The first table requiring updates is `C_ODI_PARAM` because your system calendar is populated using the ODI programs. This table is displayed as `C_ODI_PARAM_VW` on the **Manage System Configurations** screen in the Control & Tactical Center. The following settings must be updated prior to using the platform.

Table 2-3 C_ODI_PARAM Initial Settings

Scenario Name	Param Name	Configuration Guidance
SIL_DAYDIMENSION	START_DT	Start date for generating the system calendar (this is different from the fiscal calendar). Set 12+ months before the start of the planned fiscal calendar to provide adequate space for adjustments to the fiscal calendar starting period. Example: If your Fiscal start period is currently February 2020, then it would be fine to start the system calendar on `20190101`. Starting from the first day of a year ensures there are no incomplete months in the calendar.

Table 2-3 (Cont.) C_ODI_PARAM Initial Settings

Scenario Name	Param Name	Configuration Guidance
SIL_DAYDIMENSION	END_DT	End date for generating the system calendar (this is different from the fiscal calendar). Set 6-12 months beyond the expected end of the fiscal calendar to ensure the final year of that calendar does not extend beyond the available dates. Example: If your Fiscal end period is currently January 2025, then you could set the end date to `20251231`. Ending on the last day of a year ensures there are no incomplete months in the calendar.
SIL_DAYDIMENSION	WEEK_START_DT_VAL	Starting day of the week for the system calendar (1 = Sunday, 2 = Monday). The default calendar setup uses a Sunday-to-Saturday week.
GLOBAL	RI_OPTIONALLY_ENCLOSED_BY	**Note:** This parameter is deprecated in the new RAP architecture and was replaced by CTX file parameters. Set a character to use for wrapping text strings in data files, such as a quotation mark ("), to allow column delimiters to occur within the strings without causing any failures in the load process. The recommended value is "
GLOBAL	CURRENCY_CODE	Set the default currency code used when loading CSV-based fact data files if none are provided on the files themselves. Defaults to 'USD'.
GLOBAL	LANGUAGE_CODE	Default language code used by the system to load data. Do not change unless your source systems are using a non-English primary language in their database and datasets. Default=EN

Table 2-3 (Cont.) C_ODI_PARAM Initial Settings

Scenario Name	Param Name	Configuration Guidance
GLOBAL	RI_PART_DDL_CNT_LIMIT	The maximum number of partitions to create during the initial setup run. The average initial setup of the calendar may need 50k-150k partitions. The recommended value is 500000 (meaning max 500k partitions)
GLOBAL	RI_INV_HIST_DAYS	The number of days to retain a zero-balance record on inventory positions. Excessive retention of zero balances can cause batch performance issues due to high data volumes. But dropping the records too soon may be detrimental to your business reporting or analytical processes if you make use of zero-balance information. Default=91 days.
GLOBAL	RI_CLOSED_PO_HIST_DAYS	The number of days to retain closed purchase orders on the daily positional snapshots. Closed purchase orders may be important for reporting or analytical processes, but typically are not needed as they do not impact your open on-order calculations. Default=30 days.
GLOBAL	RI_GEN_PROD_RECLASS_IND	Set to 'Y' to enable RI to automatically generate item-level reclass records. Can only be used in a non-RMS implementation. Requires that full product files are sent every day, to detect when an item moves between hierarchy positions even if no other change occurred.
GLOBAL	RI_INT_ORG_DS_MANDATORY_IND	Set to 'Y' to require input data on the Organization hierarchy interface for the batch to run. This will prevent the batch from executing if the data files were not uploaded properly for a given day or the file was missing from the upload.
GLOBAL	RI_PROD_DS_MANDATORY_IND	Set to 'Y' to require input data on the Product hierarchy interface for the batch to run. This will prevent the batch from executing if the data files were not uploaded properly for a given day or the file was missing from the upload.
SIL_RETAIL_COHEADDIMENSION	RI_MIS_COHEAD_REQ_IND	Seed missing customer order (CO) head IDs from sales fact to CO Dimension. If you are providing customer order IDs on your sales history load, make sure to set this to 'Y'.

Table 2-3 (Cont.) C_ODI_PARAM Initial Settings

Scenario Name	Param Name	Configuration Guidance
SIL_RETAILCOLINEDIMENSION	RI_MIS_COLINE_REQ_IND	Seed missing customer order (CO) line IDs from sales fact to CO Dimension. If you are providing customer order line IDs on your sales history load, make sure to set this to 'Y'.
SIL_EMPLOYEEDIMENSION	RI_MIS_CASHIER_REQ_IND	Seed missing Cashier IDs from sales fact to Employee dimension. If you are providing employee IDs on your sales history load, make sure to set this to 'Y'.
SIL_RETAILCUSTOMERDIMENSION	RI_MIS_CUSTOMER_REQ_IND	Seed missing customer IDs from sales fact to Customer dimension. If you are providing customer IDs on your sales history load, make sure to set this to 'Y'.
SIL_RETAILPROMODIMENSION	RI_MIS_PROMO_REQ_IND	Seed missing promotions from the sales promo fact to the Promotion dimension. If you are providing promotion IDs on your sales history load and not providing a Promotion file, make sure to set this to 'Y'.

Retail Insights contains many additional configurations in the C_ODI_PARAM table that are not necessary for platform initialization, but may be needed for your project. This includes Merchandise Financial Planning configurations for specifying custom planning levels to be used in the integration between MFP and RI. The default parameters align with MFP's default plan outputs, but if you are customizing MFP to use a different base intersection, then you must also update those values in C_ODI_PARAM. Refer to the *Retail Insights Implementation Guide* for complete details on Planning Configurations.

W_LANGUAGES_G Initialization

The `W_LANGUAGES_G` table controls all the languages supported in the translatable database data. This applies to areas such as product names, location names, attribute values, season/phase descriptions, and other text-based descriptors. This is important mainly to Retail Insights, which supports displaying data in multiple languages in reporting and analytics. It is recommended to delete all languages from this table that will not be used because every language code in this table will have records generated for it in some interfaces, creating unnecessary data that can impact system performance.

For example, product names will automatically have database records initialized for every supported language in this configuration table, even if the data you are providing does not contain any of those languages. This creates significant amounts of data in your product descriptions table, which may not serve any real purpose for your implementation. If you are only using a single primary language, then you can safely delete all but one row from `W_LANGUAGES_G`. The default row to preserve is the one with a language code of `US` which is used for American English.

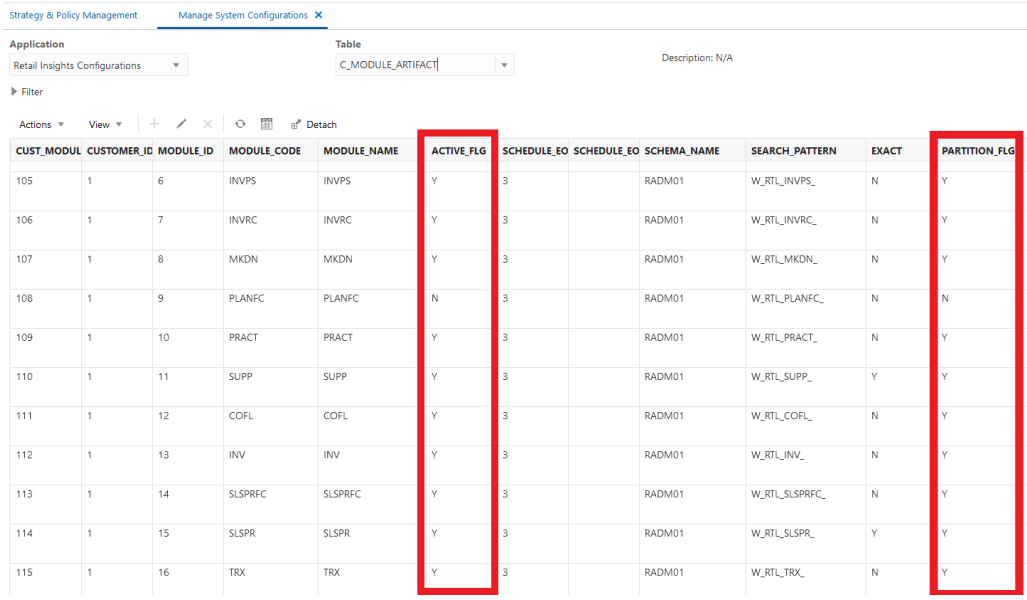
C_MODULE_ARTIFACT Initialization

The `C_MODULE_ARTIFACT` table is used by the database to configure table partitioning. Many tables in Retail Insights are partitioned based on the business calendar (usually by calendar date or fiscal week) and this partitioning must be performed immediately after the business

calendar is loaded. You should perform this step regardless of which application modules you are implementing, because all foundation data passes through this architecture.

Before running partitioning procedures, you must validate this table has all rows set to `ACTIVE_FLG=Y` and `PARTITION_FLG=Y` with the exception of `W_RTL_PLANFC*` tables (PLANFC module), which should not be partitioned at this time and must have a flag values of `N` instead.

You also must choose whether you are planning to load the Planning facts (such as `W_RTL_PLAN1_PROD1_LC1_T1_FS`) for plan/budget data in RI or RSP. If you are not using the table right away, you should also disable the PLAN modules, like PLAN1. You can revisit this setup later to perform additional partitioning as needed.



CUST_MODUL	CUSTOMER_ID	MODULE_ID	MODULE_CODE	MODULE_NAME	ACTIVE_FLG	SCHEDULE_EO	SCHEDULE_EO	SCHEMA_NAME	SEARCH_PATTERN	EXACT	PARTITION_FLG
105	1	6	INVPS	INVPS	Y	3		RADM01	W_RTL_INVPS_	N	Y
106	1	7	INVRC	INVRC	Y	3		RADM01	W_RTL_INVRC_	N	Y
107	1	8	MKDN	MKDN	Y	3		RADM01	W_RTL_MKDN_	N	Y
108	1	9	PLANFC	PLANFC	N	3		RADM01	W_RTL_PLANFC_	N	N
109	1	10	PRACT	PRACT	Y	3		RADM01	W_RTL_PRACT_	N	Y
110	1	11	SUPP	SUPP	Y	3		RADM01	W_RTL_SUPP_	Y	Y
111	1	12	COFL	COFL	Y	3		RADM01	W_RTL_COFL_	N	Y
112	1	13	INV	INV	Y	3		RADM01	W_RTL_INV_	N	Y
113	1	14	SLSPRFC	SLSPRFC	Y	3		RADM01	W_RTL_SLSPRFC_	N	Y
114	1	15	SLSPR	SLSPR	Y	3		RADM01	W_RTL_SLSPR_	Y	Y
115	1	16	TRX	TRX	Y	3		RADM01	W_RTL_TRX_	N	Y

C_MODULE_EXACT_TABLE Initialization

The `C_MODULE_EXACT_TABLE` table is used for defining flexible partitioning strategies on certain tables. Most data in this table can be left as-is, but you must update this table if you plan to load Planning or Budget information into the `W_RTL_PLAN1_PROD1_LC1_T1_FS` interface. The partition level must align with the data level of your plan (typically day or week). To configure the plan partitions, you must update the table `C_MODULE_EXACT_TABLE` where `MODULE_CODE = PLAN1`. Modify the columns `PARTITION_COLUMN_TYPE` and `PARTITION_INTERVAL` to be one of the following values:

- If your input data will be at Day level, set both columns to `DY`
- If your input data will be at Week level, set both columns to `WK`

You must then enable the partitioning process in `C_MODULE_ARTIFACT` by locating the row for `MODULE_CODE=PLAN1` and setting `ACTIVE_FLG=Y` and `PARTITION_FLG=Y`. If your plan data will extend into the future, you must also change `PARTITION_FUTURE_PERIOD` to the number of future months that need partitions built (for example, use a value of `6M` to partition 6 months into the future).

C_HIST_LOAD_STATUS

The C_HIST_LOAD_STATUS table is used to track the progress of historical loads of data, primarily inventory position and pricing facts. You may edit the following fields on this table based on your implementation needs:

- HIST_LOAD_LAST_DATE – Specifies the planned final date for the end of your historical loads (for example, the end of the 2-year period you plan to load into RAP). The history load programs will assume that you are providing each week of inventory in sequence from earliest to latest and process the data in that order.
- ENABLED_IND – Turns on or off a specific table load for historical data. Most of the tables in these processes are only required for Retail Insights, and the rest can be disabled to improve performance. Set to a value of N to disable a table load.
- MAX_COMPLETED_DATE – The load programs use this to keep track of the last loaded week of data. It does not allow you to reload this week or any prior week, so if you are trying to start over again after purging some history, you must also reset this field.
- HIST_LOAD_STATUS – The load programs uses this to track the status of each step in the load process. If your program gets stuck on invalid records change this field back to INPROGRESS before re-running the job. If you are restarting a load after erasing history data, then you need to clear this field of any values.

If you are implementing Retail Insights, then enable all INV and PRICE modules in the table (set ENABLED_IND to Y). If you are only implementing Science or Planning application modules, then the following history tables should be enabled; all others should be disabled (set ENABLED_IND to N).

- W_RTL_PRICE_IT_LC_DY_F
- W_RTL_PRICE_IT_LC_DY_HIST_TMP
- W_RTL_INV_IT_LC_DY_F
- W_RTL_INV_IT_LC_WK_A
- W_RTL_INV_IT_LC_DY_HIST_TMP

As you load data files for one or more weeks of history per run, the value of MAX_COMPLETED_DATE and HIST_LOAD_STATUS automatically update to reflect the progress you have made. If you need to restart the process (for example, you have loaded test data and need to start over with production data) these two columns must first be cleared of all data before beginning the history load again.

C_SOURCE_CDC

The C_SOURCE_CDC table is used for changed data capture (CDC) parameters for the integrations between the Retail Insights data warehouse and the Planning application schemas. In general, this table is updated automatically as batches are run. However, it is important to know when you may need to modify these values.

For most interfaces, the table will initially have no records. The first time an integration batch program runs, it will take all the data from the source table and move it to the export table. It will then create a C_SOURCE_CDC record for the target table name, with a value for LAST_MIN_DATE and LAST_MAX_DATE matching the timeframe extracted. On the next run, it will look at LAST_MAX_DATE as the new minimum extract date and pulls data greater than that date

from the source table. If you are performing history loads for tables, such as Sales Transactions, you may need to change these dates if you have to re-send data to Planning for past periods.

Specifically for positional data (at this time only Inventory Position), the usage is not quite the same. Positional data will always send the current end-of-week values to Planning, it does not look at historical weeks as part of the normal batch process. A separate historical inventory integration program is provided in an ad hoc process, which will allow you to send a range of weeks where `LAST_MIN_DATE` is the start of the history you wish to send, and `LAST_MAX_DATE` is the final date of history before normal batches take it forward. It is common to load inventory from end to end in isolation as it is a data-intensive and time-consuming process to gather, load, and validate inventory positions for multiple years of history.

W_GLOBAL_CURR_G

The W_GLOBAL_CURR_G table is used by Retail Insights to support up to three additional currencies in reporting and aggregation (other fields above 3 are not used at this time). RI pre-populates global currency fields in all aggregation tables based on the specified currency codes. The desired codes are added to one row in this table and must align with the Exchange Rates data provided separately. This table is available from the Control & Tactical Center and is not a required configuration for any project unless you wish to report on additional currencies in Retail Insights.

Example data to be inserted to this table:

DATASOURCE_NUM_ID	TENANT_ID	GLOBAL1_CURR_CODE	GLOBAL2_CURR_CODE	GLOBAL3_CURR_CODE	GLOBAL1_RATE_TYPE	GLOBAL2_RATE_TYPE	GLOBAL3_RATE_TYPE	DEFAULT_LOC_RATE_TYPE
1	DEFAULT	INR	AED	PEN	Corporate	Corporate	Corporate	Corporate

Application Configurations

In addition to the platform configurations defined above, each application on the platform has its own system and runtime options that need to be reviewed and updated. The information below will guide you to the appropriate content for each application's configuration options.

Retail Insights

Retail Insights has a significant number of configurations, primarily in the `C_ODI_PARAM` table, which controls batch processes and reporting behaviors throughout the application. If you are implementing Retail Insights as part of your project, review the "Setup and Configuration" chapter of the *Retail Insights Implementation Guide*.

Science Platform and Forecasting

Each Science application has parameters that are specific to the batch processing, data movement, algorithms, and user interfaces of those modules. These configurations are stored in several database tables available through the Control &

Tactical Center. If you are implementing any Science applications as part of your project, review the *Retail Science Cloud Services Implementation Guide*.

If you are implementing Merchandise Financial Planning, then you are required to configure and use the Forecasting module in the Science application interface. This requires initial configurations to select forecast parameters, as well as post-data load configurations to select forecast data levels and perform testing of the chosen algorithm. For basic information about Forecasting and what the Science application functionality can support, refer to the "Manage Forecast Configurations" section in the *Science User Guide*.

To configure the forecast process for MFP, use the **Manage System Configurations** screen in the Control Center to review and modify the configurations in RSE_CONFIG. These values can be set up now, but you cannot complete the rest of the forecasting process until your foundation data has been loaded into the Science platform.

Appl Code	Parameter Name	Description
RSE	EXTENDED_HIERARCHY_SRC	Data source providing extended hierarchy data using either `RMS` or `NON-RMS`. Default value is `NON-RMS`. If using RMFCS or RMS-sourced data, must update this parameter.
RSE	LOAD_EXTENDED_PROD_HIER	`Y` or `N` value. This parameter is used by the product hierarchy ETL to know whether the extended product hierarchy is needed. Default value is `Y`. If the product hierarchy data had 9 levels, keep this value as `Y`. If it has 7 levels, change this value to `N`.
PMO	PMO_PROD_HIER_TYPE	The hierarchy ID to use for the Offer Optimization product. Default value is `3`. If the product hierarchy data has 9 levels (that is, it has an extended hierarchy), keep this value as `3`. If it has 7 levels (extended hierarchy is unneeded), change this value to `1`.
RSE	PROD_HIER_SLSTXN_HIER_LEVEL_ID	This parameter identifies the hierarchy level at which sales transactions are provided (7-Style, 8-Style/color or 9 Style/color/Size). It MUST match the extended hierarchy leaf level. Default value is `9`.
PMO	PMO_AGGR_INVENTORY_DATA_FLG	Specifies whether inventory data is present and if it should be used when aggregating activities data. Set this value to `N` if inventory data is not loaded (inventory data is not required for MFP forecasting but it is required for other applications like Offer Optimization, Inventory Optimization and Retail Demand Forecasting). Default value is `Y`.

Planning Platform

Planning Applications such as MFP (Merchandise Financial Planning) can be set up using the Planning Platform (RPAS CE). It allows customers to use a Standard GA template version

or configurable planning solution versions. Refer to the Planning application-specific Implementation Guides for more details about these options.

3

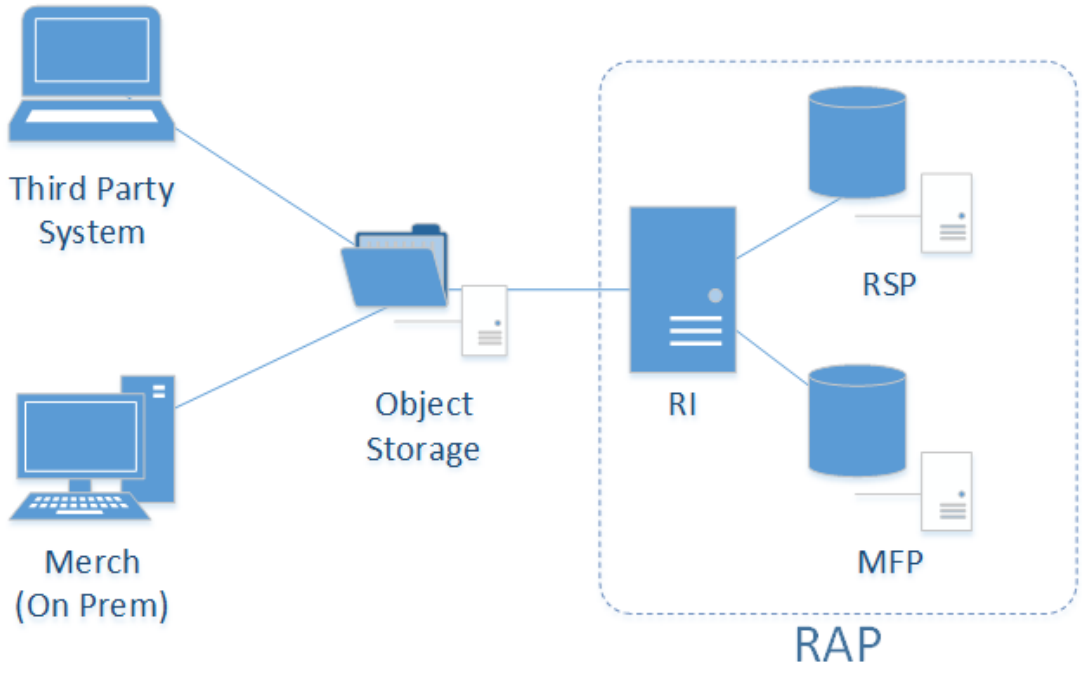
Data Loads and Initial Batch Processing

This chapter describes the data requirements for implementing modules of the Retail Analytics Platform, where to get additional information for optional or application-specific data interfaces, and how to load an initial dataset into the cloud environments and distribute it across your desired applications.

Data Requirements

Preparing data for one or more modules of the Retail Analytics Platform can consume a significant amount of project time, so it is crucial to identify the minimum data requirements for the platform first, followed by additional requirements that are specific to your implementation plan. Data requirements that are called out for the platform are typically shared across all modules, meaning you only need to provide the inputs once to leverage them everywhere. This is the case for foundational data elements, such as your product and location hierarchies. Foundation data must be provided for any module of the platform to be implemented. Foundation data is provided using different sources depending on your current software landscape, including the on-premise Oracle Retail Merchandising System (RMS) or 3rd-party applications.

Figure 3-1 Inbound Foundation Data Flows



Application-specific data requirements are in addition to the shared foundation data, and may only be used by one particular module of the platform. These application data requirements

may have different formats and data structures from the core platform-level dataset, so pay close attention to those additional interface specifications. References and links are provided later in this chapter to guide you to the relevant materials for application-specific inputs and data files.

If you are using RMS as your primary data source, then you may not need to produce some or all of these foundation files, as they will be created by other Oracle Retail processes for you. However, it is often the case that historical data requires a different set of foundation files from your future post-implementation needs. If you are loading manually-generated history files, or you are not using an Oracle Retail data source for foundation data, then review the rest of this section for details.

Platform Data Requirements

There is a subset of core platform data files that can be created and loaded once and then used across some or all application modules. These files use a specific format as detailed below.

> **Note:**
>
> Every application on the Retail Analytics Platform has additional data needs beyond this foundation data. But this common set of files can be used to initialize the system before moving on to those specific requirements

The first table defines the minimum dimensional data. A dimension is a collection of descriptive elements, attributes, or hierarchical structures that provide context to your business data. Dimensions tell the platform what your business looks like and how it operates. This is not the entire list of possible dimension files, just the main ones needed to use the platform. Refer to Legacy Foundation File Reference for a complete list of available platform foundation files, along with a cross-reference to the legacy interfaces they most closely align with. A complete interface specification document is also available in My Oracle Support to assist you in planning your application-specific interface needs.

Table 3-1 Common Foundation Dimensions

Dimension	Filename(s)	Usage
Product	PRODUCT.csv	The product foundation data includes the items you sell, their core merchandise attributes, and their hierarchical structure in your source systems.
Organization	ORGANIZATION.csv	The organization foundation data includes all of your business entities involved in the movement or sale of merchandise. This includes stores, warehouses, partner/finisher locations, web stores, and virtual warehouses. It also includes your organizational hierarchy and core location attributes.

Table 3-1 (Cont.) Common Foundation Dimensions

Dimension	Filename(s)	Usage
Calendar	CALENDAR.csv	The calendar foundation data defines your business (or fiscal) calendar. This is the calendar that you operate in when making critical business decisions, reporting on financial results, and planning for the future. The most common calendar used by all modules of the platform is a 4-5-4 fiscal calendar.
Exchange Rates	EXCH_RATE.csv	Exchange rates define the conversion of monetary values from the currency they are recorded in to your primary operating currency. Most data is provided to the platform in the local currency of the data source, and it is converted to your primary currency during the load process.
Product Attributes	ATTR.csv PROD_ATTR.csv	Product attributes describe the physical and operational characteristics of your merchandise and are a critical piece of information for many Science modules, such as Demand Transference and Size Profile Optimization. They are not required as an initial input to start data loads but will eventually be needed for most applications to function.
System Parameters	RA_SRC_CURR_PARAM_G.dat	Parameters file that supports certain batch processes, such as the ability to load multiple ZIP files and run batches in sequence. Include this file with nightly RI batch uploads to specify the current business date, which enables the system to run multiple batches in sequence without customer input. Required once you begin nightly or weekly batch uploads.

The other set of foundation files are referred to as facts. Fact data covers all of the actual events, transactions, and activities occurring throughout the day in your business. Each module in the platform has specific fact data needs, but the most common ones are listed below. At a minimum, you should expect to provide **Sales**, **Inventory**, and **Receipts** data for use in most platform modules. The intersection of all data (meaning which dimensional values are used) is at a common level of **item/location/date**. Additional identifiers may be needed on some files; for example, the sales data should be at the transaction level, the inventory file has a clearance indicator, and the adjustments file has type codes and reason codes.

Table 3-2 Common Foundation Facts

Dimension	Filename(s)	Usage
Sales	SALES.csv	Transaction-level records for customer sales (wholesale data provided separately). Used across all modules.

Table 3-2 (Cont.) Common Foundation Facts

Dimension	Filename(s)	Usage
Inventory	INVENTORY.csv	Physical inventory levels for owned merchandise as well as consignment and concession items. Used across all modules.
Receipts	RECEIPT.csv	Inventory receipts into any location, including purchase order and transfer receipts. Used by Insights, Planning, and the Science modules to identify the movement of inventory into a location and to track first/last receipt date attributes.
Adjustments	ADJUSTMENT.csv	Inventory adjustments made at a location, including shrink, wastage, theft, stock counts, and other reasons. Used by Insights and Planning modules.
Purchase Orders	ORDER_HEAD.csv ORDER_DETAIL.csv	The purchase order data for all orders placed with suppliers. Held at a level of order number, supplier, item, location, and date. Separate files are needed for the order attributes and order quantities/ amounts. Used by Insights and Planning modules.
Markdowns	MARKDOWN.csv	The currency amount above or below the listed price of an item when that item's price is modified for any reason (planned markdown, POS discount, promotional sale, and so on). Used By Insights and Planning modules.
Transfers	TRANSFER.csv	The movement of inventory between two locations (both physical and virtual). Used By Insights and Planning modules.
Returns to Vendor	RTV.csv	The return of owned inventory to a supplier or vendor. Used by Insights and Planning modules.
Prices	PRICE.csv	The current selling retail value of an item at a location. Used by Insights and Science modules.
Costs	COST.csv	The base unit cost and derived net costs of an item at a location. Used by Insights and Science modules.
Wholesale/Franchise	SALES_WF.csv	Sales and markdown data from wholesale and franchise operations. Used by all modules.
Deal Income	DEAL_INCOME.csv	Income associated with deals made with suppliers and vendors. Used by Insights and Planning modules.

Details on which application modules make use of specific files (or columns within a file) can be found in the *Interfaces Guide* on My Oracle Support. Make sure you have a full understanding of the data needs for each application you are implementing before moving on to later steps in the process. If it is your first time creating these files,

read Data File Generation, for important information about key file structures and business rules that must be followed for each foundation file.

File Upload Samples

When you first upload foundation data into the platform, you will likely provide a small subset of the overall set of files required by your applications. The following examples show a possible series of initial file uploads to help you verify that you are providing the correct sets of data. All dimension files in initialization and history loads must be full snapshots of data; never send partial or incremental files.

Example #1: Calendar Initialization

When you first configure the system you must upload and process the `CALENDAR.csv` file. You will generate the file following the specifications, and also provide a context (ctx) file, as described in Data File Generation.

File to upload: `RIHIST_RMS_DATA.zip`

Zip file contents:

- `CALENDAR.csv`
- `CALENDAR.csv.ctx` (or `W_MCAL_PERIOD_DTS.dat.ctx` on legacy architecture)

Example #2: Product and Location Setup

You have finalized the calendar and want to initialize your core product and organization dimensions. You must provide the `PRODUCT.csv` and `ORGANIZATION.csv` files along with their context files, as described in Data File Generation.

File to upload: `RIHIST_RMS_DATA.zip`

Zip file contents:

- `PRODUCT.csv`
- `PRODUCT.csv.ctx` (or `W_PRODUCT_DTS.dat.ctx` on legacy architecture)
- `ORGANIZATION.csv`
- `ORGANIZATION.csv.ctx` (or `W_INT_ORG_DTS.dat.ctx` on legacy architecture)

Example #3: Full dimension load

You have a set of dimension files you want to process using the initial dimension load ad hoc processes in POM.

File to upload: `RIHIST_RMS_DATA.zip`

Zip file contents:

- `PRODUCT.csv`
- `PRODUCT.csv.ctx` (or `W_PRODUCT_DTS.dat.ctx` on legacy architecture)
- `ORGANIZATION.csv`
- `ORGANIZATION.csv.ctx` (or `W_INT_ORG_DTS.dat.ctx` on legacy architecture)

- ATTR.csv
- ATTR.csv.ctx (or W_ATTR_DTS.dat.ctx on legacy architecture)
- PROD_ATTR.csv
- PROD_ATTR.csv.ctx (or W_PRODUCT_ATTR_DTS.dat.ctx on legacy architecture)
- EXCH_RATE.csv
- EXCH_RATE.csv.ctx (or W_EXCH_RATE_DTS.dat.ctx on legacy architecture)

Example #4: Sales Data Load

You have finished the dimensions and you are ready to start processing sales history files.

File to upload: RIHIST_RMS_DATA.zip

Zip file contents:

- SALES.csv
- SALES.csv.ctx (or W_RTL_SLS_TRX_IT_LC_DY_FTS.dat.ctx on legacy architecture)

Example #5: Multi-File Fact Data Load

Once you are confident in your data file format and contents, you may send multiple files as separate ZIP uploads for sequential loading in the same run. This process uses a numerical sequence on the end of the ZIP file name.

Files to upload: RIHIST_RMS_DATA.zip.1, RIHIST_RMS_DATA.zip.2, RIHIST_RMS_DATA.zip.3

Zip file contents (in each uploaded zip):

- SALES.csv
- SALES.csv.ctx (or W_RTL_SLS_TRX_IT_LC_DY_FTS.dat.ctx on legacy architecture)

Uploading ZIP Packages

When providing data to the platform, push the compressed files into Object Storage using a ZIP file format. Review the Object Storage section for details on how to interact with Object Storage. The ZIP file you use will depend on the data you are attempting to load. The standard ZIP packages include:

Table 3-3 Platform ZIP File Usage

Filenames	Frequency	Notes
RIHIST_RMS_DATA.zip	Ad Hoc	Used for: • Historical files, such as sales and inventory history for the last 1-2 years. • Loading initial dimensions, such as calendar, merchandise, and location hierarchies prior to history loads. • Initial seeding loads.
RIHIST_RMS_DATA.zip.1RIHIST_RMS_DATA.zip.2...RIHIST_RMS_DATA.zip.N	Ad Hoc	Multiple zip uploads are supported for sending historical fact data which should be loaded sequentially. Append a sequence number on the ZIP files starting from 1 and increasing to N, where N is the number of files you are loading.
RAP_DATA.zip RI_RMS_DATA.zip RI_CE_DATA.zip RI_MFP_DATA.zip RI_EXT_DATA.zip	Daily	Can be used for daily ongoing loads into Retail Insights, and for any daily data going to downstream applications through RI's nightly batch. Different files can be used for different source systems.
ORASE_WEEKLY_ADHOC.zip	Ad Hoc	Used for loading RSP files with ad hoc processes.
ORASE_WEEKLY.zip	Weekly	Used for weekly batch files sent directly to RSP.
ORASE_INTRADAY.zip	Intraday	Used for intraday batch files sent directly to RSP.

Other supported file packages, such as output files and optional input files, are detailed in each module's implementation guides. Except for Planning-specific integrations and customizations (which support additional integration paths and formats), it is expected that all files will be communicated to the platform using one of the filenames above.

Important

> **Important:**
>
> Once you begin loading historical data with `RIHIST_RMS_DATA.zip` uploads, do not keep re-processing the same dimension file for the same business date. Also, do not run the ad hoc process for a dimension without a file present, as you are then effectively closing out all data and telling the system you no longer have any active products or locations for that date. After successfully loading a dimension and moving on to other files, it is recommended to disable the related POM jobs from your process until you need it again.

Preparing to Load Data

Implementations can follow this general outline for the data load process:

1. Initialize the business and system calendars and perform table partitioning, which prepares the database for loading fact data.
2. Begin staging dimension files, but do NOT load them all the way through to the target tables. For the first few runs, it is best to leverage APEX or DV to validate your data in the staging tables and ensure it is correct and complete before loading it the rest of the way through.
3. Load verified dimension data into the target tables, and perform additional checks on the data from DV/APEX or using RI reports.
4. If implementing any Science or Planning module, load the dimension data to those systems now. Data might work fine within RI but have issues only visible after processing in those systems. Don't start loading history data if your dimensions are not working with all target applications.
5. Load the first set of history files (for example, one month of sales or inventory) the rest of the way. Again pause at the staging layer, validate the data with SQL queries, and then proceed to load it into RI.
6. If implementing any Science or Planning module, stop here and load the history data to those systems as well. Validate that the history data in those systems is complete and accurate per your business requirements.
7. Continue loading history data into RAP until you are finished with all data. You can stop at any time to move some of the data into downstream modules for validation purposes.
8. After history loads are complete, all positional tables, such as Inventory Position, need to be seeded with a full snapshot of source data before they can be loaded using regular nightly batches. This seeding process is used to create a starting position in the database which can be incremented by daily delta extracts.
9. When all history and seeding loads are completed and downstream systems are also populated with that data, nightly batches can be started.

Before you begin this process, it is best to prepare your working environment by identifying the tools and connections needed for all your Oracle cloud services that will allow you to interact with the platform, as detailed in Implementation Tools and Data File Generation.

Prerequisites for loading files and running POM processes include:

Prerequisite	Tool / Process
Upload ZIPs to Object Store	File Transfer Service (FTS) scripts
Invoke adhoc jobs to unpack and load the data	Postman (or similar REST API tool)
Monitor job progress after invoking POM commands	POM UI (Batch Monitoring tab)
Monitoring data loads	APEX / DV (direct SQL queries)

Users must also have the necessary permissions in Oracle Identity Cloud Services (IDCS) or Oracle Cloud Infrastructure Identity and Access Management (OCI IAM) to perform all the implementation tasks. Before you begin, ensure that your user has at least the following groups (or their `_PREPROD` equivalent):

Access Needed	Groups Needed
Batch Job Execution	BATCH_ADMINISTRATOR_JOB PROCESS_SERVICE_ADMIN_JOB
Database Monitoring	DVContentAuthor (DV) DATA_SCIENCE_ADMINISTRATOR_JOB (APEX)
Retail Home	RETAIL_HOME_ADMIN PLATFORM_SERVICES_ADMINISTRATOR
RI and RSP Configurations	ADMINISTRATOR_JOB
MFP Configurations	MFP_ADMIN

Calendar and Partition Setup

This is the first step that must be performed in all new environments, including projects that will not be implementing RI, but only RSP or MFP. Before beginning this step, ensure your configurations are complete per the initial configuration sections in the prior chapter. Your `START_DT` and `END_DT` variables must be set correctly for your calendar range (`START_DT` must be earlier than the first date in your calendar file) and the `C_MODULE_ARTIFACT` table must have all of the required tables enabled for partitioning. `C_MODULE_EXACT_TABLE` must be configured if you need PLAN partitions for planning data loads.

If this is the first time running any batch programs through POM, then there are some important, one-time updates that are necessary before you begin this process:

- Ensure that the POM program for `SETUP_CTX_FILE_JOB` is enabled in the **Administration** tab in both Nightly and Standalone schedules for all processes where it occurs.
- Restart your POM schedules from the **Monitoring** tab and execute the `SETUP_CTX_FILE_PROCESS_ADHOC` process to verify it is successful.
- If you encounter an error when running this job, open a Service Request (SR) with Oracle Support, as a configuration step may be missing in your environment.

> **Note:**
>
> These steps cannot be skipped and the error while running this job cannot be ignored. If this process is not able to run successfully, then your CTX files will be ignored during data loads. Once these processes are successful, continue on to the steps below.

1. Upload the calendar file `CALENDAR.csv` (and associated context file) through Object Storage or SFTP (packaged using the `RIHIST_RMS_DATA.zip` file).
2. Execute the **`HIST_ZIP_FILE_LOAD_ADHOC`** process. Example Postman message body:

```
{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"HIST_ZIP_FILE_LOAD_ADHOC"
}
```

3. Verify that the two jobs in the ZIP file load process completed successfully using the POM **Monitoring** screen. Download logs for the tasks as needed for review.
4. Execute the **CALENDAR_STG_CSV_LOAD_ADHOC** process. Verify the data from your file was loaded to the target tables W_MCAL_PERIOD_DTS and W_MCAL_PERIOD_DS using APEX. Refer to Sample Validation SQLs for sample queries you can use for this.
5. Execute the **CALENDAR_LOAD_ADHOC** process. This transforms the data and moves it into all internal RI tables.
6. Validate all jobs in the calendar process are successful in POM. Validate that the tables W_MCAL_PERIOD_D and W_MCAL_DAY_D are populated with a full range of calendar dates based on your uploaded file. Do not begin the partitioning process until you are confident the calendar data is correctly loaded, as invalid calendar data can result in incorrect partitions.

   If you need to reload the same file multiple times due to errors, you must Restart the Schedule in POM and then run the ad hoc process C_LOAD_DATES_CLEANUP_ADHOC before repeating these steps. This will remove any load statuses from the prior run and give you a clean start on the next execution. When you are finished loading data, move on to the partitioning process below.

   Begin the partitioning process using the Postman message below. There are two date parameters provided for this command:

   a. The 1st date value specifies the first day of partitioning. It must be before the first actual day of data being loaded, but you should not create excessive partitions for years of data you won't be using, as it can impact performance of the system.
   b. The 2nd date (ETL business date) specifies the target business date, which is typically the day the system should be at after loading all history data and starting daily batches. It is okay to guess some date in the future for this value, but note that the partition process automatically extends 4 months past your specified date.

```
{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"CREATE_PARTITION_ADHOC",
"requestParameters":"jobParams.CREATE_PARTITION_PRESETUP_
JOB=2018-12-30,jobParams.ETL_BUSINESS_DATE_JOB=2021-02-06"
}
```

The partitioning process will take some time (~5 hours per 100k partitions) to complete if you are loading multiple years of history, as this may require 100,000+ partitions to be created across the data model. This process must be completed successfully before continuing with the data load process. Contact Oracle Support if there are any questions or concerns. Partitioning can be performed after some data has been loaded; however, it will take significantly longer to execute, as it has to move all of the loaded data into the proper partitions.

If your historical time range is very long, the partitioning process may require multiple executions. It has the ability to run multiple times and resume the partitioning from where it left off. An easy way to see if all partitions are created is to run the process and see if it completes right away (in under a minute). If the process is still taking a long time to run on the second or third execution, then it hasn't completed all partitions yet.

You can also estimate the number of partitions needed based on the details below:

- RAP needs to partition around 120 week-level tables, so take the number of weeks in your history time window multiplied by this number of tables.
- RAP needs to partition around 160 day-level tables, so take the number of days in your history time window multiplied by this number of tables.

For a 3-year history window, this results in: 120*52*3 + 160*365*3 = 193,920 partitions. If you wish to confirm your counts before proceeding, you can execute these queries from APEX:

```
select count(*) cnt from dba_tab_partitions where table_owner = 'RADM01' and
table_name like 'W_RTL_%'
select table_name, count (*) cnt from dba_tab_partitions where table_owner =
'RADM01' and table_name like 'W_RTL_%' group by table_name
```

The queries should return a count roughly equal to your expected totals (it will not be exact, as the data model will add/remove tables over time and some tables come with pre-built partitions or default `MAXVALUE` partitions).

Loading Data from Files

When history and initial seed data comes from flat files, use the following tasks to upload them into RAP:

Table 3-4 Flat File Load Overview

Activity	Description
Initialize Dimensions	Initialize dimensional data (products, locations, and so on) to provide a starting point for historical records to join with. Separate initial load processes are provided for this task.
Load History Data	Run history loads in one or multiple cycles depending on the data volume, starting from the earliest date in history and loading forward to today.
Reloading Dimensions	Reload dimensional data as needed throughout the process to maintain correct key values for all fact data. Dimensional files can be provided in the same package with history files and ad hoc processes run in sequence when loading.
Seed Positional Facts	Seed initial values for positional facts using full snapshots of all active item/locations in the source system. This must be loaded for the date prior to the start of nightly batches to avoid gaps in ongoing data.
Run Nightly Batches	Nightly batches must be started from the business date after the initial seeding was performed.

Completing these steps will load all of your data into the Retail Insights data model, which is required for all implementations. From there, proceed with moving data downstream to other applications as needed, such as Science modules and Merchandise Financial Planning.

> **Note:**
>
> All steps are provided sequentially, but can be executed in parallel. For example, you may load dimensions into RI, then on to RSP and MFP before loading any historical fact data. While historical fact data is loaded, other activities can occur in MFP such as the domain build and configuration updates.

Initialize Dimensions

Loading Dimensions into RI

You cannot load any fact data into the platform until the related dimensions have been processed and verified. The processes in this section are provided to initialize the core dimensions needed to begin fact data loads and verify file formats and data completeness. Some dimensions which are not used in history loads are not part of the initialization process, as they are expected to come in the nightly batches at a later time.

For the complete list of dimension files and their file specifications, refer to the *Retail Insights Interfaces Guide* on My Oracle Support. The steps below assume you have enabled or disabled the appropriate dimension loaders in POM per your requirements. The process flow examples also assume CSV file usage, different programs are available for legacy DAT files. The *RI and Science Operations Guide* provides a list of all the job and process flows used by foundation data files, so you can identify the jobs required for your files and disable unused programs in POM.

1. Provide your dimension files and context files through Object Storage (packaged using the `RIHIST_RMS_DATA.zip` file). All files should be included in a single zip file upload.
2. Execute the `HIST_ZIP_FILE_LOAD_ADHOC` process.
3. Execute the `LOAD_CURRENT_BUSINESS_DATE_ADHOC` process to set the load date to one day before the start of your history load timeframe. Sample Postman message body:

```
{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"LOAD_CURRENT_BUSINESS_DATE_ADHOC",
"requestParameters":"jobParams.ETL_BUSINESS_DATE_JOB=2017-12-31"
}
```

4. Execute the `LOAD_DIM_INITIAL_CSV_ADHOC` process, which imports the files to pre-staging tables with no transformations applied. These table names end in DTS or FTS, and you can query them using APEX at this point. Refer to Sample Validation SQLs for sample queries you can use for this.
5. Execute the `LOAD_EXT_DIM_INITIAL_SI_ADHOC` processes to stage data into the core data model, applying transformations as needed to format the data. This moves the data to various tables ending in `DS` or `DS_TL`. Validate your data before

proceeding. Refer to Sample Validation SQLs for sample queries you can use for this.

6. Execute the LOAD_DIM_INITIAL_ADHOC process to load your dimension data from the staging tables. The ETL date on the command should be the same one used in the prior steps.

```
{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"LOAD_DIM_INITIAL_ADHOC",
"requestParameters":"jobParams.ETL_BUSINESS_DATE_JOB=2017-12-31"
}
```

If any jobs fail during this load process, you may need to alter one or more dimension data files, re-send them in a new zip file upload, and re-execute the programs. Only after all core dimension files have been loaded (such as PRODUCT, ORGANIZATION, and EXCH_RATE) can you proceed to history loads for fact data.

If you need to reload the same file multiple times due to errors, you must Restart the Schedule in POM and then run the ad hoc process C_LOAD_DATES_CLEANUP_ADHOC before repeating these steps. This will remove any load statuses from the prior run and give you a clean start on the next execution.

> **Note:**
>
> You must not reload the same foundation dimension file on the same business date if it was already imported successfully. Certain file changes, like moving items or locations into different hierarchy levels, can only happen once per business date. Change the ETL business date parameter before reloading a dimension file which might have hierarchy changes in it.

As a best practice, you should disable all POM jobs in the LOAD_DIM_INITIAL_ADHOC process except the ones you are providing new files for. This will prevent any mistakes, such as running the Organization loads without having a file present (which can result in closing all records in the database). Once you have loaded data successfully through an interface, do not re-run that interface again, unless you have advanced the business date and provided a new file.

Loading Dimensions to Other Applications

Once you have successfully loaded dimension data, you may choose to pause the dataload process and push the dimensions to the Science Platform or Merchandise Financial Planning (if applicable). This allows for parallel data validation and domain build activities to occur while you continue loading data. Review sections Sending Data to Science and Sending Data to Planning for details on the POM jobs you may execute for this.

The main benefits of this order of execution are:

1. Validating the hierarchy structure from the Science interface provides an early view for the customer to see some application screens with their data.
2. MFP performs the domain build activity without waiting for history file loads to complete, and can start to do other planning implementation activities in parallel to the history loads.

3. Data can be made available for custom development or validations in Innovation Workbench.

Load History Data

Historical fact data is a core foundational element to all solutions in the Retail Analytics Platform. As such, this phase of the implementation can take the longest amount of time during the project, depending on the volumes of data, the source of the data, and the amount of transformation and validation that needs to be completed before and after loading it into the platform.

Before loading any large history files such as sales and inventory, update the multi-threading parameter (`LOC_NUM_OF_THREAD`) in the `C_ODI_PARAM` table for these processes. In general, a good starting value for multi-threading is 8 threads, after which time you can monitor the job performance of each thread and add or remove threads as needed. Thread-level runtimes can be found on the `C_LOAD_DATES` table, which is viewable from APEX or the Tactical & Control Center.

The scenario names below are most often used in the history load processes and should have multi-threading values checked/adjusted in `C_ODI_PARAM`.

> **Note:**
>
> All related jobs MUST use the same thread value, or data could be missed. For example, all `INVPOSITION` scenarios must use the same value.

```
SIL_RETAIL_SALESTRANSACTIONFACT
SIL_RETAILINVPOSITIONFACT
SIL_RETAILINVRECEIPTSFACT
PLP_RETAILINVPOSITIONITLCGMHAGGREGATE
PLP_RETAILINVPOSITIONITLCWKAGGREGATE
PLP_RETAILINVPOSITIONSCLCDYCURRRECLASSAGGREGATE
PLP_RETAILINVPOSITIONSCLCDYWKCURRRECLASSAGGREGATE
PLP_RETAILSALESITLCDYTEMPLOAD
```

It is also important to know where in the RAP database you can look to find what data has been processed, what data may have been rejected or dropped due to issues, and how far along in the overall load process you are. The following tables provide critical pieces of information throughout the history load process.

Table 3-5 Inbound Load Status Tables

Table	Usage
C_HIST_LOAD_STATUS	Tracks the progress of historical ad hoc load programs for inventory and pricing facts. This table will tell you which Retail Insights tables are being populated with historical data, the most recent status of the job executions, and the most recently completed period of historical data for each table. Use APEX or Data Visualizer to query this table after historical data load runs to ensure the programs are completing successfully and processing the expected historical time periods.

Table 3-5 (Cont.) Inbound Load Status Tables

Table	Usage
C_LOAD_DATES	Check for detailed statuses of historical load jobs. This is the only place that tracks this information at the individual ETL thread level. For example, it is possible for an historical load using 8 threads to successfully complete 7 threads but fail on one thread due to data issues. The job itself may just return as Failed in POM, so knowing which thread failed will help identify the records that may need correcting and which thread should be reprocessed.
W_ETL_REJECTED_RECORDS	Summary table capturing rejected fact record counts that do not get processed into their target tables in Retail Insights. Use this to identify other tables with specific rejected data to analyze. Does not apply to dimensions, which do not have rejected record support at this time.
E$_W_RTL_SLS_TRX_IT_LC_DY_TMP	Example of a rejected record detail table for Sales Transactions. All rejected record tables start with the `E$_` prefix. These tables are created at the moment the first rejection occurs for a load program. `W_ETL_REJECTED_RECORDS` will tell you which tables contain rejected data for a load.

When loading data from flat files for the first time, it is common to have bad records that cannot be processed by the RAP load procedures, such as when the identifiers on the record are not present in the associated dimension tables. The foundation data loads leverage rejected record tables to capture all such data so you can see what was dropped by specific data load and needs to be corrected and reloaded. These tables do not exist until rejected records occur during program execution. Periodically monitor these tables for rejected data which may require reloading.

Inventory Position History Load

RI supports the loading of inventory position history using full, end-of-week snapshots. These weekly snapshots may be provided one week at a time or as multiple weeks in a single file. The data must be loaded sequentially from the earliest week to the latest week with no gaps or out-of-order periods. For example, you cannot start with the most recent inventory file and go backward; you must start from the first week of history. Refer to Data File Generation for more details on the file requirements.

A variety of `C_ODI_PARAM` settings are available to disable inventory features that are not required for your implementation. All of the following parameters can be changed to a value of `N` during the history load and enabled later for daily batches, as it will greatly improve the load times:

- `RI_INVAGE_REQ_IND` – Disables calculation of first/last receipt dates and inventory age measures. Receipt date calculation is used in RI and also Offer Optimization (as a method of determining entry/exit dates for items). It is also used for Forecasting for the Short Lifecycle (SLC) methods.
- `RA_CLR_LEVEL` – Disables the mapping of clearance event IDs to clearance inventory updates. Used only in RI reporting.
- `RI_PRES_STOCK_IND` – Disables use of replenishment data for presentation stock to calculate inventory availability measures. Used only in RI reporting.

- RI_BOH_SEEDING_IND – Disables the creation of initial beginning-on-hand records so analytics has a non-null starting value in the first week. Used only in RI reporting.
- RI_MOVE_TO_CLR_IND – Disables calculation of move-to-clearance inventory measures when an item/location goes into or out of clearance status. Used only in RI reporting.
- RI_MULTI_CURRENCY_IND – Disables recalculation of primary currency amounts if you are only using a single currency. Should be enabled for multi-currency, or disabled otherwise.

The following steps describe the process for loading inventory history:

1. If you need inventory to keep track of First/Last Receipt Dates for use in Offer Optimization or Forecasting (SLC) then you must first load a RECEIPT.csv file for the same historical period as your inventory file (because it is used in forecasting that may make it required for your Inventory Optimization loads as well, if you plan to use SLC forecasting). You must also set RI_INVAGE_REQ_IND to Y. Receipts are loaded using the process HIST_CSV_INVRECEIPTS_LOAD_ADHOC. Receipts may be provided at day or week level depending on your history needs.
2. Create the file INVENTORY.csv containing one or more weeks of inventory snapshots in chronological order along with your CTX file to define the columns that are populated. The DAY_DT value on every record must be an end-of-week date (Saturday by default). The only exception to this is the final week of history, which may be the middle of the week, as long as you perform initial seeding loads on the last day of that week.
3. Upload the history file and its context file to Object Storage using the RIHIST_RMS_DATA.zip file.
4. Update column HIST_LOAD_LAST_DATE on the table C_HIST_LOAD_STATUS to be the date matching the last day of your overall history load (will be later than the dates in the current file). This can be done from the Control & Tactical Center.
5. Execute the HIST_ZIP_FILE_LOAD_ADHOC process.
6. Execute the HIST_STG_CSV_INV_LOAD_ADHOC process to stage your data into the database. Validate your data before proceeding. Refer to Sample Validation SQLs for sample queries you can use for this.
7. Execute the HIST_INV_LOAD_ADHOC batch process to load the file data. The process loops over the file one week at a time until all weeks are loaded. It updates the C_HIST_LOAD_STATUS table with the progress, which you can monitor from APEX or DV. Sample Postman message bodies:

```
{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"HIST_STG_CSV_INV_LOAD_ADHOC"
}

{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"HIST_INV_LOAD_ADHOC"
}
```

This process can be repeated as many times as needed to load all history files for the inventory position. Remember that inventory cannot be loaded out of order, and you cannot go back in time to reload files after you have processed them. If you load a set of inventory files and then find issues during validation, erase the tables in the database and restart the load with corrected files.

If your inventory history has invalid data, you may get rejected records and the batch process will fail with a message that rejects exist in the data. If this occurs, you cannot proceed until you resolve your input data, because rejections on positional data MUST be resolved for one date before moving onto the next. If you move onto the next date without reprocessing any rejected data, that data is lost and cannot be loaded at a later time without starting over. When this occurs:

1. The inventory history load will automatically populate the table `W_RTL_REJECT_DIMENSION_TMP` with a list of invalid dimensions it has identified. If you are running any other jobs besides the history load, you can also run the process `W_RTL_REJECT_DIMENSION_TMP_ADHOC` to populate that table manually. You have the choice to fix the data and reload new files or proceed with the current file
2. After reviewing the rejected records, run `REJECT_DATA_CLEANUP_ADHOC`, which will erase the `E$` table and move all rejected dimensions into a skip list. The list is loaded to the table `C_DISCARD_DIMM`. Skipped identifiers will be ignored for the current file load, and then reset for the start of the next run.
3. If you want to fix your files instead of continuing the current load, stop here and reload your dimensions and/or fact data following the normal process flows.
4. To reset the inventory load either for resuming the current run or starting over with a new file, update the `C_HIST_LOAD_STATUS` table for the current load to set the `HIST_LOAD_STATUS` back to `INPROGRESS` for two tables: the one it failed on, and `W_RTL_INV_IT_LC_DY_HIST_TMP`.
5. If you are resuming with the current file with the intent to skip all data in `C_DISCARD_DIMM`, restart the POM job now.

Log a Service Request with Oracle Support for assistance with any of the above steps if you are having difficulties with loading inventory history or dealing with rejected records.

Sales History Load

RI supports the loading of sales transaction history using actual transaction data or daily/weekly sales totals. If loading data at an aggregate level, all key columns (such as the transaction ID) are still required to have some value. The sales data may be provided for a range of dates in a single file. The data should be loaded sequentially from the earliest week to the latest week but, unlike inventory position, you may have gaps or out-of-order loads, because the data is not stored positionally.

If you are not loading sales history for Retail Insights specifically, then there are many aggregation programs that can be disabled in the POM standalone process. These programs populate additional tables used only in BI reporting. The following list of jobs can be disabled if you are not using RI (effectively all aggregate “A” tables and jobs except for the item/location/week level can be turned off):

- W_RTL_SLS_IT_LC_DY_A_JOB
- W_RTL_SLS_IT_LC_GMH_A_JOB
- W_RTL_SLS_SC_LC_DY_A_JOB
- W_RTL_SLS_SC_LC_WK_A_JOB

- W_RTL_SLS_CL_LC_DY_A_JOB
- W_RTL_SLS_CL_LC_WK_A_JOB
- W_RTL_SLS_DP_LC_DY_A_JOB
- W_RTL_SLS_DP_LC_WK_A_JOB
- W_RTL_SLS_IT_DY_TMP_JOB
- W_RTL_SLS_IT_DY_A_JOB
- W_RTL_SLS_IT_WK_A_JOB
- W_RTL_SLS_SC_DY_A_JOB
- W_RTL_SLS_SC_WK_A_JOB
- W_RTL_SLS_LC_DY_TMP_JOB
- W_RTL_SLS_LC_DY_A_JOB
- W_RTL_SLS_LC_WK_A_JOB
- W_RTL_SLS_IT_LC_DY_SN_TMP_JOB
- W_RTL_SLS_IT_LC_DY_SN_A_JOB
- W_RTL_SLS_IT_LC_WK_SN_A_JOB
- W_RTL_SLS_IT_DY_SN_A_JOB
- W_RTL_SLS_IT_WK_SN_A_JOB
- W_RTL_SLS_SC_LC_DY_CUR_TMP_JOB
- W_RTL_SLS_SC_LC_DY_CUR_A_JOB
- W_RTL_SLS_SC_LC_WK_CUR_A_JOB
- W_RTL_SLS_CL_LC_DY_CUR_A_JOB
- W_RTL_SLS_DP_LC_DY_CUR_A_JOB
- W_RTL_SLS_CL_LC_WK_CUR_A_JOB
- W_RTL_SLS_DP_LC_WK_CUR_A_JOB
- W_RTL_SLS_SC_DY_CUR_A_JOB
- W_RTL_SLS_CL_DY_CUR_A_JOB
- W_RTL_SLS_DP_DY_CUR_A_JOB
- W_RTL_SLS_SC_WK_CUR_A_JOB
- W_RTL_SLS_CL_WK_CUR_A_JOB
- W_RTL_SLS_DP_WK_CUR_A_JOB
- W_RTL_SLSPR_PC_IT_LC_DY_A_JOB
- W_RTL_SLSPR_PP_IT_LC_DY_A_JOB
- W_RTL_SLSPR_PE_IT_LC_DY_A_JOB
- W_RTL_SLSPR_PP_CUST_LC_DY_A_JOB
- W_RTL_SLSPR_PC_CS_IT_LC_DY_A_JOB
- W_RTL_SLSPR_PC_HH_WK_A_JOB

After confirming the list of enabled sales jobs, perform the following steps:

1. Create the file `SALES.csv` containing one or more days of sales data along with a CTX file defining the columns which are populated.
2. Upload the history files to Object Storage using the `RIHIST_RMS_DATA.zip` file.
3. Execute the `HIST_ZIP_FILE_LOAD_ADHOC` process.
4. Execute the `HIST_STG_CSV_SALES_LOAD_ADHOC` process to stage the data in the database. Validate your data before proceeding. Refer to Sample Validation SQLs for sample queries you can use for this.
5. Execute the `HIST_SALES_LOAD_ADHOC` batch processes to load the data. If no data is available for certain dimensions used by sales, then the load process can seed the dimension from the history file automatically. Enable seeding for all of the dimensions according to the initial configuration guidelines; providing the data in other files is optional.

> **Note:**
>
> Several supplemental dimensions are involved in this load process, which may or may not be provided depending on the data requirements. For example, sales history data has promotion identifiers, which would require data on the promotion dimension.

Sample Postman message bodies:

```
{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"HIST_STG_CSV_SALES_LOAD_ADHOC"
}

{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"HIST_SALES_LOAD_ADHOC"
}
```

This process can be repeated as many times as needed to load all history files for the sales transaction data. If you are sending data to multiple RAP applications, do not wait until all data files are processed to start using those applications. Instead, load a month or two of data files and process them into all apps to verify the flows before continuing.

> **Note:**
>
> Data cannot be reloaded for the same records multiple times, as sales data is treated as additive in Retail Insights. If data needs correction, you must post only the delta records (for example, send -5 to reduce a value by 5 units). Create a Service Request with Oracle Support for assistance with clearing out incorrect or unwanted data in your environment.

Price History Load

Certain applications, such as Promotion and Markdown Optimization, require price history to perform their calculations. Price history is similar to inventory in that it is a positional, but it can be loaded in a more compressed manner due to the extremely high data volumes involved. The required approach for price history is as follows:

1. Update `C_HIST_LOAD_STATUS` for the `PRICE` records in the table, specifying the last date of history load, just as you did for inventory.
2. Create an initial, full snapshot of price data in `PRICE.csv` for the first day of history and load this file into the platform using the history processes in this section. All initial price records must come with a type code of 0.
3. Create additional `PRICE` files containing just price changes for a period of time (such as a month) with the appropriate price change type codes and effective day dates for those changes. Load each file one at a time using the history processes.
4. The history procedure will iterate over the provided files day by day, starting from the first day of history, up to the last historical load date specified in `C_HIST_LOAD_STATUS` for the pricing fact. For each date, the procedure checks the staging data for effective price change records and loads them, then moves on to the next date.

The process to perform price history loads is similar to the inventory load steps. It uses the `PRICE.csv` file and the `HIST_CSV_PRICE_LOAD_ADHOC` process (the price load only has one load process instead of two like sales/inventory). Just like inventory, you must load the data sequentially; you cannot back-post price changes to earlier dates than what you have already loaded. Refer to Data File Generation for complete details on how to build this file.

Other History Loads

While sales and inventory are the most common facts to load history for, you may also want to load history for other areas such as receipts and transfers. Separate ad hoc history load processes are available for the following fact areas:

- 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

All of these interfaces deal with transactional data (not positional) so you may use them at any time to load history files in each area.

> **Note:**
>
> These processes are intended to support history data for downstream applications such as Science and Planning, so the tables populated by each process by default should satisfy the data needs of those applications. Jobs not needed by those apps are not included in these processes.

Reloading Dimensions

It is common to reload dimensions at various points throughout the history load, or even in-sync with every history batch run. Ensure that your core dimensions, such as the product and location hierarchies, are up-to-date and aligned with the historical data being processed. To reload dimensions, you may follow the same process as described in the Initial Dimension Load steps, ensuring that the current business load date in the system is on or before the date in history when the dimensions will be required. For example, if you are loading history files in a monthly cadence, ensure that new product and location data required for the next month has been loaded no later than the first day of that month, so it is effective for all dates in the history data files. Do not reload dimensions for the same business date multiple times, advance the date to some point after the prior load each time you want to load new sets of changed dimension files.

It is also very important to understand that history load procedures are unable to handle reclassifications that have occurred in source systems when you are loading history files. For example, if you are using current dimension files from the source system to process historical data, and the customer has reclassified products so they are no longer correct for the historical time periods, then your next history load may place sales or inventory under the new classifications, not the ones that were relevant in history. For this reason, reclassifications should be avoided if at all possible during history load activities, unless you can maintain historical dimension snapshots that will accurately reflect historical data needs.

Seed Positional Facts

Once sales and inventory history have been processed, you will need to perform seeding of the positional facts you wish to use. Seeding a fact means to load a full snapshot of the data for all active item/locations, thus establishing a baseline position for every possible record before nightly batches start loading incremental updates to those values. Seeding of positional facts should only occur once history data is complete and daily batch processing is ready to begin.

Initial seed loads can take a long time to execute, as each file contains all possible item/location combinations that are active in the source systems. Plan for 2-3 days to execute these processes until you have performed them once and have a better estimate for the total time required. Seed loads should also be done for a week-ending date, so that you do not have a partial week of daily data in the system when you start daily batches.

If you did not previously disable the optional inventory features in `C_ODI_PARAM` (parameters `RI_INVAGE_REQ_IND`, `RA_CLR_LEVEL`, `RI_PRES_STOCK_IND`, `RI_BOH_SEEDING_IND`, `RI_MOVE_TO_CLR_IND`, and `RI_MULTI_CURRENCY_IND`) then you should review these settings now and set all parameters to “N” if the functionality is not required. Once this is done, follow the steps below to perform positional seeding:

1. Create the files containing your initial full snapshots of positional data. It may be one or more of the following:

- PRICE.csv
- COST.csv
- INVENTORY.csv
- ORDER_DETAIL.csv (ORDER_HEAD.csv should already be loaded using dimension process)

2. Upload the files to Object Storage using the RIHIST_RMS_DATA.zip file.
3. Execute the LOAD_CURRENT_BUSINESS_DATE_ADHOC process to set the load date to be the next week-ending date after the final date in your history load.

```
{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"LOAD_CURRENT_BUSINESS_DATE_ADHOC",
"requestParameters":"jobParams.ETL_BUSINESS_DATE_JOB=2017-12-31"
}
```

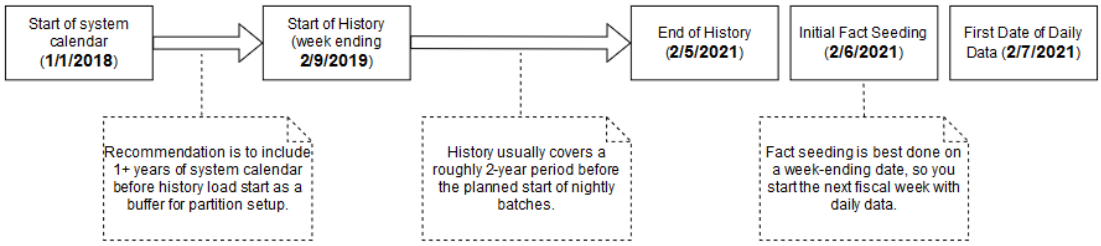
4. Execute the ad hoc seeding batch processes depending on which files have been provided. Sample Postman messages:

```
{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"SEED_CSV_W_RTL_PRICE_IT_LC_DY_F_PROCESS_ADHOC"
}

{
"cycleName": "Adhoc",
"flowName":"Adhoc",
"processName":"SEED_CSV_W_RTL_NCOST_IT_LC_DY_F_PROCESS_ADHOC"
}

{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"SEED_CSV_W_RTL_BCOST_IT_LC_DY_F_PROCESS_ADHOC"
}

{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"SEED_CSV_W_RTL_INV_IT_LC_DY_F_PROCESS_ADHOC"
}

{
"cycleName":"Adhoc",
"flowName":"Adhoc",
"processName":"SEED_CSV_W_RTL_PO_ONORD_IT_LC_DY_F_PROCESS_ADHOC"
}
```

Once all initial seeding is complete and data has been validated, you are ready to perform a regular batch run. Provide the data files expected for a full batch, such as RAP_DATA.zip or RI_RMS_DATA.zip for foundation data, RI_MFP_DATA.zip for planning

data (for RI reporting and RSP forecasting), and any Science Platform files using the `ORASE_WEEKLY.zip` files. If you are sourcing daily data from RMS then you need to ensure that the RDE batch flow is configured to run nightly along with the RAP batch schedule. Batch dependencies between RDE and RI should be checked and enabled, if they are not already turned on.

From this point on, the nightly batch takes care of advancing the business date and loading all files, assuming that you want the first load of nightly data to occur the day after seeding. The following diagram summarizes a potential set of dates and activities using the history and seeding steps described in this chapter:



> **Note:**
>
> The sequential nature of this flow of events must be followed for positional facts (for example, inventory) but not for transactional facts (such as sales). Transactional data supports posting for dates other than what the current system date is, so you can choose to load sales history at any point in this process.

Run Nightly Batches

As soon as initial seeding is performed, prepare for starting nightly batch runs in full. Nightly batch schedules can be configured in parallel with the history load processes using a combination of the Customer Modules Management (in Retail Home) and the POM administration screens. It is not recommended to configure the nightly jobs manually in POM, as there are over 500 batch programs; choosing which to enable can be a time-intensive and error-prone activity. Customer Modules Management greatly simplifies this process and preserves dependencies and required process flows. Batch Orchestration describes the batch orchestration process and how you can configure batch schedules for nightly execution.

Once you move to nightly batches, you may also want to switch interfaces from Full to Incremental loading of data. Several RI interfaces, such as the Product dimension, can be loaded incrementally, sending only the changed records every day instead of a full snapshot. These options use the `IS_INCREMENTAL` flag in the `C_ODI_PARAM` table and can be accessed from the Control & Tactical Center. If you are unsure of which flags you want to change, refer to the *Retail Insights Implementation Guide* for detailed descriptions of all parameters.

As part of nightly batch uploads, ensure that the parameter file `RA_SRC_CURR_PARAM_G.dat` is included in each ZIP package, and that it is being automatically updated with the current business date for that set of files.

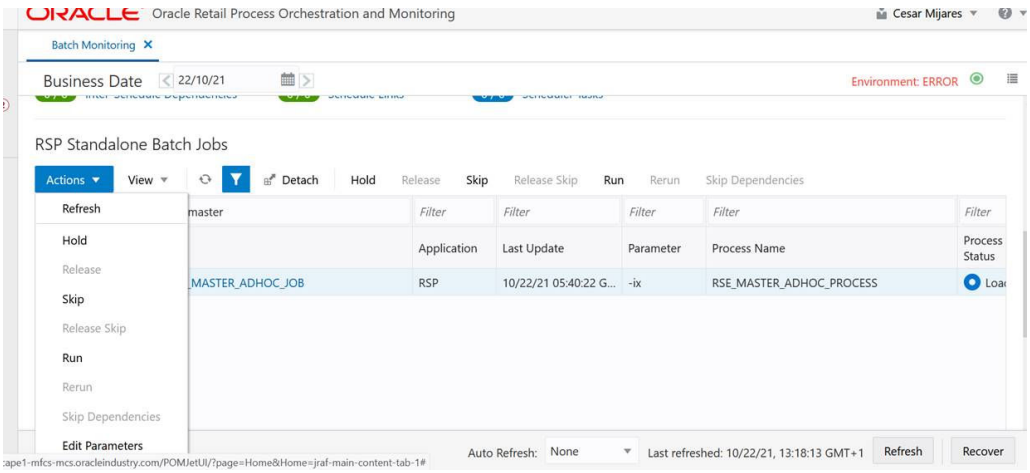
Sending Data to Science

All Science modules leverage a common batch infrastructure to initialize the core dataset, followed by ad hoc, application-specific programs to generate additional data as needed. Before loading any data into a Science module, it is necessary to complete initial dimension loads into RI and validate that core structures (calendar, products, locations) match what you expect to see. You may also perform some or all of the history loads before extracting any data to Science. Once you are comfortable with the data that has been loaded in, leverage the following jobs to move data into one or more Science applications.

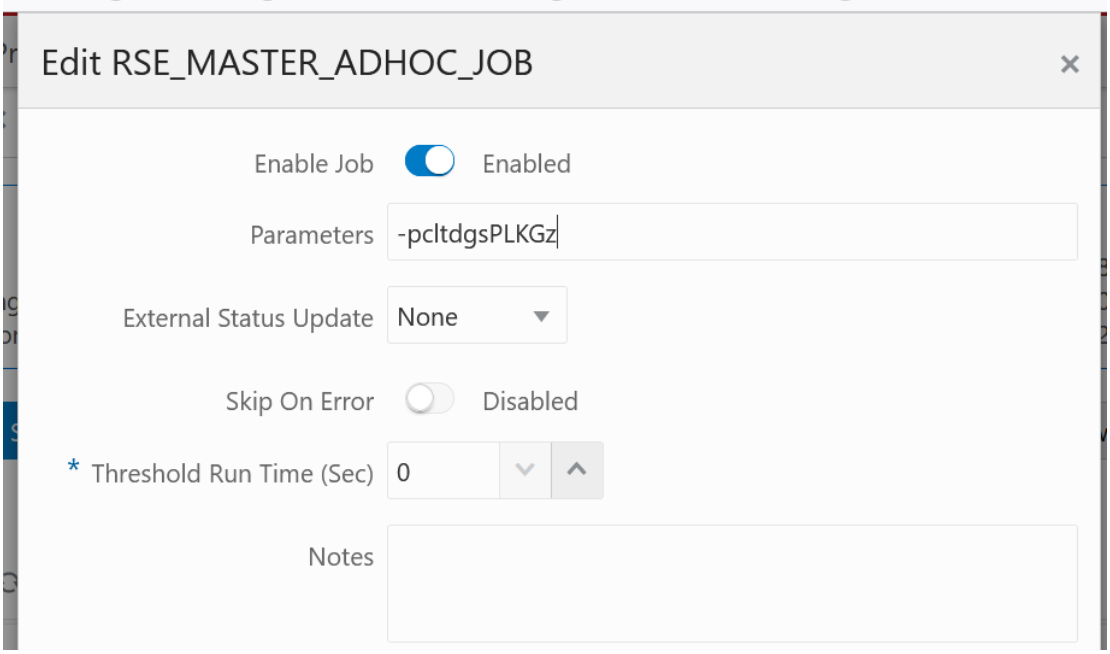
Table 3-6 Extracts for Science

POM Process Name	Usage Details
INPUT_FILES_ADHOC_PROCESS	Receive inbound zip files intended for Science, archive and extract the files. This process looks for the `ORASE_WEEKLY_ADHOC.zip` file.
RSE_MASTER_ADHOC_PROCESS	Foundation data movement from RI to Science, including core hierarchies and dimensional data. Accepts many different parameters to run specific steps in the load process.
<app>_MASTER_ADHOC_PROCESS	Each Science module, such as SPO or IO, has a master job for extracting and loading data that is required for that application, in addition to the `RSE_MASTER` processes.

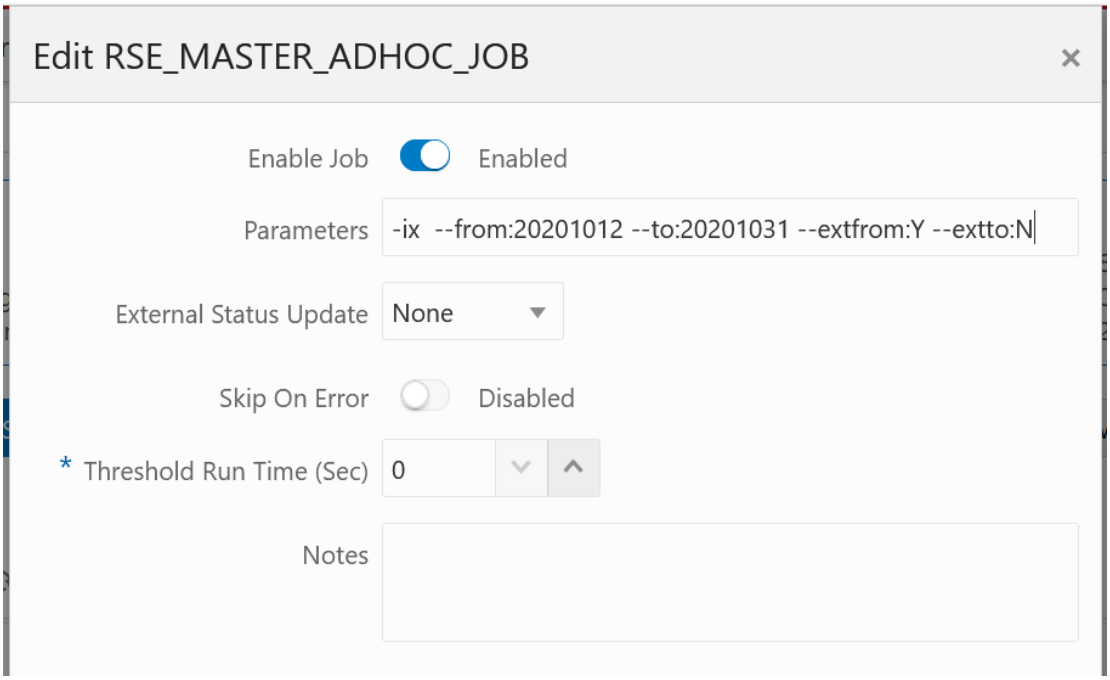
Because Science Platform ad hoc procedures have been exposed using only one job in POM, they are not triggered like RI procedures. Science programs accept a number of single-character codes representing different steps in the data loading process. These codes can be provided directly in POM by editing the Parameters of the job in the Batch Monitoring screen, then executing the job through the user interface.



For example, this string of parameters will move all dimension data from RI to RSP:



Additional parameters are available when moving periods of historical data, such as inventory and sales:



A typical workflow for moving core foundation data into Science is:

1. Load the core foundation files (like Calendar, Product, and Organization) into RI.
2. Use the `RSE_MASTER_ADHOC_PROCESS` to move those same datasets to RSP, providing specific flag values to only run the needed steps.

3. Load some of your history files for Sales to validate the inputs.
4. Load the same range of sales to RSP using the sales load with optional from/to date parameters.
5. Repeat the previous two steps until all sales data is loaded into both RI and RSP.

   Performing the process iteratively provides you early opportunities to find issues in the data before you've loaded everything, but it is not required. You can load all the data into RSP at one time.

Follow the same general flow for the other application-specific, ad hoc flows into the Science modules. For a complete list of parameters in each program, refer to the RI and Science Operations Guide.

Sending Data to Planning

If a Planning module is being implemented, then additional RI jobs should be executed as part of the initial and nightly batch runs. These jobs are available through ad hoc calls, and the nightly jobs are included in the RI nightly schedule. Review the list below for more details on the core Planning extracts available.

Table 3-7 Extracts for Planning

POM Job Name	Usage Details
W_PDS_PRODUCT_D_JOB	Exports a full snapshot of Product master data (for non-pack items only) and associated hierarchy levels.
W_PDS_ORGANIZATION_D_JOB	Exports a full snapshot of Location master data (for stores and warehouses only) and associated hierarchy levels.
W_PDS_CALENDAR_D_JOB	Exports a full snapshot of Calendar data at the day level and associated hierarchy levels. **Note:** While RI exports the entire calendar, PDS will only import 5 years around the `RPAS_TODAY` date (current year +/- 2 years).
W_PDS_EXCH_RATE_G_JOB	Exports a full snapshot of exchange rates.
W_PDS_DIFF_D_JOB	Exports a full snapshot of Differentiators such as Color and Size.

Table 3-7 (Cont.) Extracts for Planning

POM Job Name	Usage Details
W_PDS_UDA_D_JOB	Exports a full snapshot of User-Defined Attributes.
W_PDS_DEALINC_IT_LC_WK_A_JOB	Incremental extract of deal income data (transaction codes 6 and 7 from RMFCS) posted in the current business week.
W_PDS_PO_ONORD_IT_LC_WK_A_JOB	Incremental extract of future on-order amounts for the current business week, based on the expected OTB date.
W_PDS_INV_IT_LC_WK_A_JOB	Incremental extract of inventory positions for the current business week. Inventory is always posted to the current week, there are no back-posted records.
W_PDS_SLS_IT_LC_WK_A_JOB	Incremental extract of sales transactions posted in the current business week (includes back-posted transactions to prior transaction dates).
W_PDS_INVTSF_IT_LC_WK_A_JOB	Incremental extract of inventory transfers posted in the current business week (transaction codes 30, 31, 32, 33, 37, 38 from RMFCS).
W_PDS_INVRC_IT_LC_WK_A_JOB	Incremental extract of inventory receipts posted in the current business week. Only includes purchase order receipts (transaction code 20 from RMFCS).
W_PDS_INVRTV_IT_LC_WK_A_JOB	Incremental extract of returns to vendor posted in the current business week (transaction code 24 from RMFCS).
W_PDS_INVADJ_IT_LC_WK_A_JOB	Incremental extract of inventory adjustments posted in the current business week (transaction codes 22 and 23 from RMFCS).
W_PDS_SLSWF_IT_LC_WK_A_JOB	Incremental extract of wholesale and franchise transactions posted in the current business week (transaction codes 82, 83, 84, 85, 86, 88 from RMFCS).
W_PDS_MKDN_IT_LC_WK_A_JOB	Incremental extract of markdowns posted in the current business week (transaction codes 11, 12, 13, 14, 15, 16, 17, 18 from RMFCS).
W_PDS_INV_IT_LC_WK_A_INITIAL_JOB	Initial history extract for inventory position data based on the timeframe established in C_SOURCE_CDC.

The PDS jobs are linked with several ad hoc processes in POM, providing you with the ability to extract specific datasets on-demand as you progress with history and initial data loads. The table below summarizes the ad hoc processes, which can be called using the standard methods such as cURL or Postman.

Table 3-8 RI Ad Hoc Processes for Planning

POM Process Name	Usage Details
LOAD_PDS_DIMENSION_PROCESS_ADHOC	Groups all of the dimension (D/G table) extracts for PDS into one process for ad hoc execution. Disable individual jobs in POM to run only some of them.
LOAD_PDS_FACT_PROCESS_ADHOC	Groups all of the fact (F/A table) extracts for PDS into one process for ad hoc execution. Disable individual jobs in POM to run only some of them.
LOAD_PDS_FACT_INITIAL_PROCESS_ADHOC	History extract process for inventory positions, run only to pull a full snapshot of inventory from RI for one or multiple weeks of data. You must set the start and end dates for extraction in the C_SOURCE_CDC table before running this process.
C_LOAD_DATES_CLEANUP_ADHOC	Purge the successful job records from C_LOAD_DATES, which is necessary when you are running the same jobs multiple times per business date.

All of the PDS fact jobs leverage the configuration table C_SOURCE_CDC to track the data that has been extracted. On the first run of any incremental process, the job will extract all available data in a single run. From that point forwards, the extract will incrementally load only the data which has been added or modified since the last extract. For the initial inventory job, it requires start/end dates to extract, so you must update C_SOURCE_CDC from the Tactical & Control Center. For all other jobs, the extract dates are written to the C_SOURCE_CDC alongside the extract table name after each execution and can be overwritten as needed when doing multiple loads or re-running the same time period. If you run the same process more than once, use C_LOAD_DATES_CLEANUP_ADHOC to reset the run statuses before the next run.

Table 3-9 Planning Integration Configuration Tables

Table	Usage
C_SOURCE_CDC	Configuration and tracking table that shows the interfaces supported for RI to Planning integration and the currently processed date ranges.
RAP_INTF_CONFIG	Configuration and tracking table for integration ETL programs between all RAP modules. Contains their most recent status, run ID, and data retention policy.
RAP_INTF_RUN_STATUS	RAP integration run history and statuses.
RAP_LOG_MSG	RAP integration logging table, specific contents will vary depending on the program and logging level.

After the planning data has been extracted from RI, the Planning systems use the same processes to extract both initial and incremental data for each interface. As long

as RI has exported historical data for the current run, the regular interfaces scheduled to run using POM will pull the historical or incremental data specific to that interface into PDS.

> **Note:**
>
> Before loading dimension data into Planning, you must configure the value of `RPAS_TODAY` and do any other initial configurations as specified in the MFP or RPAS CE Implementation Guides.

Generating Forecasts

Before you can complete an MFP implementation, you must set up and run forecasting within the Science platform. Review the steps below for the initial setup of Forecasting:

1. In **Manage Forecast Configurations** in the RSP UI, start by setting up a run type in the Setup train stop.
2. Click the **+** icon above the table and fill in the fields in the popup. For MFP forecasting, the forecast method should be selected as **Automatic Exponential Smoothing**.
3. Create a run type for each forecast measure/forecast intersection combination that is required for MFP.
4. Create test runs in the Test train stop once you are done setting up the run types:
    a. Click a run type in the top table, then click on the **+** icon in the bottom table to create a run.
    b. If necessary, change the configurations parameters for the estimation process and forecast process in their respective tabs in the **Create Run** popup.

       For example, if you want to test a run using Bayesian method, edit the **Estimation Method** parameter in the **Estimation** tab using the edit icon above the table.
    c. After modifying and reviewing the configuration parameters, click the **Submit** button to start the run.
5. Once the run is complete, the status changes to **Forecast Generation Complete**.

Doing test runs is an optional step. In addition to that, you will need to modify and review the configurations of the run type, activate the run type, enable auto-approve and map the run type to the downstream application (in this case to MFP). In the Manage train stop, select a row, click **Edit Configurations Parameters** and edit the estimation and forecast parameters as needed. Once you are done, go to **Review** tab, click **Validate**, then close the tab.

> **Note:**
>
> If the run type is active, you will only be able to view the parameters. To edit the parameters, the run type must be inactive.

To activate the run type and enable the auto-approve, select a run type in the table and click the corresponding buttons above the table. Lastly, to map the run type to MFP, go to the Map train stop and click the **+** icon to create a new mapping.

When configuring forecasts for the MFP base implementation, the following list of forecast runs may be required, and you will want to configure and test each run type following the general workflow above. Additional runs can be added to satisfy your MFP implementation requirements.

> **Note:**
>
> The “Channel” level in MFP is often referred to as “Area” level in RI and RSP, so be sure to select the correct levels which align to your hierarchy.

MFP Plan	MFP Levels	Method	Data Source	Measure
MFP Merch Target	Department-Channel-Week	Auto ES	Store Sales	Regular and Promotion Gross Sales Amt
MFP Merch Target	Department-Channel-Week	Auto ES	Store Sales	Clearance Gross Sales Amt
MFP Merch Target	Department-Channel-Week	Auto ES	Store Sales	Regular and Promotion Gross Sales Unit
MFP Merch Target	Department-Channel-Week	Auto ES	Store Sales	Clearance Gross Sales Unit
MFP Merch Target	Department-Channel-Week	Auto ES	Store Sales	Total Returns Amount
MFP Merch Target	Department-Channel-Week	Auto ES	Store Sales	Total Returns Units
MFP Merch Plan	Subclass-Channel-Week	Auto ES	Store Sales	Regular and Promotion Gross Sales Amt
MFP Merch Plan	Subclass-Channel-Week	Auto ES	Store Sales	Clearance Gross Sales Amt
MFP Merch Plan	Subclass-Channel-Week	Auto ES	Store Sales	Regular and Promotion Gross Sales Unit
MFP Merch Plan	Subclass-Channel-Week	Auto ES	Store Sales	Clearance Gross Sales Unit
MFP Merch Plan	Subclass-Channel-Week	Auto ES	Store Sales	Total Returns Amount
MFP Merch Plan	Subclass-Channel-Week	Auto ES	Store Sales	Total Returns Units
MFP Location Target	Company-Location-Week	Auto ES	Store Sales	Total Gross Sales Amount
MFP Location Target	Company-Location-Week	Auto ES	Store Sales	Total Gross Sales Unit
MFP Location Target	Company-Location-Week	Auto ES	Store Sales	Total Returns Amount
MFP Location Target	Company-Location-Week	Auto ES	Store Sales	Total Returns Units
MFP Location Plan	Department-Location-Week	Auto ES	Store Sales	Total Gross Sales Amount

MFP Plan	MFP Levels	Method	Data Source	Measure
MFP Location Plan	Department-Location-Week	Auto ES	Store Sales	Total Gross Sales Unit
MFP Location Plan	Department-Location-Week	Auto ES	Store Sales	Total Returns Amount
MFP Location Plan	Department-Location-Week	Auto ES	Store Sales	Total Returns Units

Migrate Data Between Environments

The process of moving all your data from one cloud environment to another is referred to as a “lift-and-shift” process. It is common to perform all of the dataload activities in this chapter in one environment, then have the final dataset copied into another environment (instead of reloading it all from the beginning). This activity is performed by Oracle and can be requested using a Service Request. The lift-and-shift process can take several days to complete, depending on data volumes and how many other Oracle applications may be involved. It is expected that you will raise SRs to schedule the activity at least 2-3 weeks in advance of the target date.

When requesting the activity, you may specify if you need to migrate the entire database (both data and structures) or only the data. The first time doing this process must be a full migration of data and structures to synchronize the source and target environments. It is currently recommended to begin your implementation in the Production environment to avoid needing a lift-and-shift in order to go live. Around the time of your go-live date, you can request a lift-and-shift be done into your non-production environments to synchronize the data for future use.

> **Note:**
>
> The product versions between the source and target must be aligned. It is the project team’s responsibility to plan for appropriate upgrades and lift-and-shift activities such that this will be true.

4

Batch Orchestration

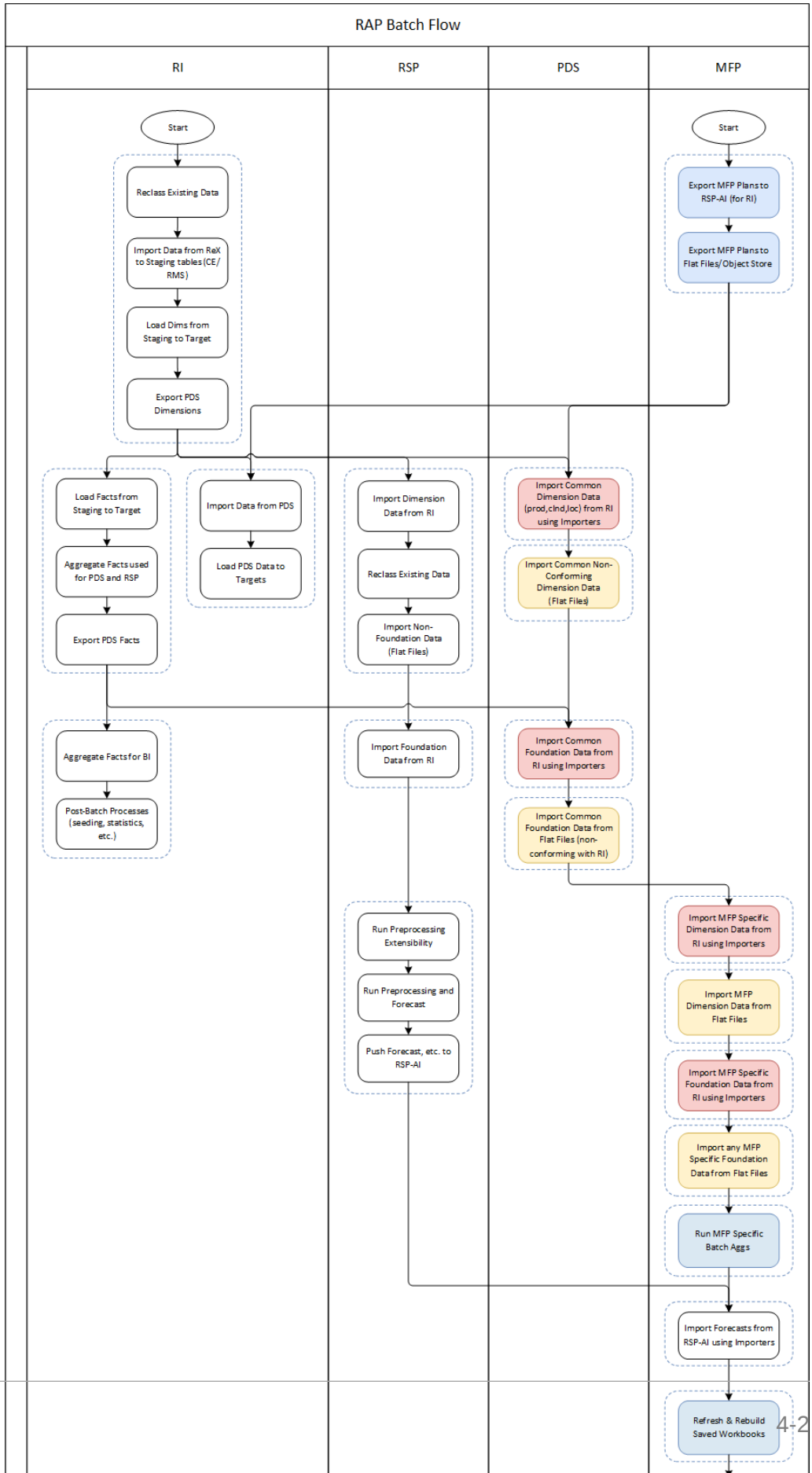
This chapter describes the tools, processes, and implementation considerations for configuring and maintaining the batch schedules used by the Retail Analytics Platform. This includes nightly, weekly, and ad hoc batch cycles added in the Process Orchestration and Monitoring (POM) tool for each of the RAP applications.

Overview

All applications on the Retail Analytics Platform have either a nightly or weekly batch schedule. Periodic batches allow the applications to move large amounts of data during off-peak hours. They can perform long-running calculations and analytical processes that cannot be completed while users are in the system, and close out the prior business day in preparation for the next one.

To ensure consistency across the platform, all batch schedules have some level of interdependencies established, where jobs of one application require processes from another schedule to complete successfully before they can begin. The flow diagram below provides a high-level view of schedule dependencies and process flows across RAP modules.

Figure 4-1 Batch Process High-Level Flow



The frequency of batches will vary by application. However, the core data flow through the platform must execute nightly. This includes data extraction from RMS by way of RDE (if used) and data loads into Retail Insights.

Downstream applications from Retail Insights, such as Merchandise Financial Planning, may only execute the bulk of their jobs on a weekly basis. This does not mean the schedule itself can run weekly (as MFP batch has been run in previous versions); those end-of-week processes now rely on consumption and transformations of data happening in nightly batches. For example, Retail Insights consumes sales and inventory data on a daily basis. However, the exports to Planning (and subsequent imports in those applications) are only run at the end of the week, and are cumulative for all the days of data up to that point in the week.

For this reason, assume that most of the data flow and processing that is happening within the platform will happen every day and plan your file uploads and integrations with non-Oracle systems accordingly.

While much of the batch process has been automated and pre-configured in POM, there are still several activities that need to be performed which are specific to each implementation. TheTable 4-1 table summarizes these activities and the reasons for doing them. Additional details will be provided in subsequent sections of this document

Table 4-1 Common Batch Orchestration Activities

Activity	Description
Initial Batch Setup	By default, most batch processes are enabled for all of the applications. It is the implementer's responsibility to disable batches that will not be used by leveraging the Customer Modules Management screen in Retail Home.
Configure POM Integrations	The POM application supports external integration methods including external dependencies and process callbacks. Customers that leverage non-Oracle schedulers or batch processing tools may want to integrate POM with their existing processes.
Schedule the Batches	Schedules in POM must be given a start time to run automatically. Once started, you have a fixed window of time to provide all the necessary file uploads, after which time the batch will fail due to missing data files.
Batch Flow Details	It is possible to export the batch schedules from POM to review process/job mappings, job dependencies, inter-schedule dependencies, and other details. This can be very useful when deciding which processes to enable in a flow or when debugging fails at specific steps in the process, and how that impacts downstream processing.

Initial Batch Setup

As discussed in Implementation Tools, setting up the initial batches requires access to two tools: POM and Retail Home. Refer to that chapter if you have not yet configured your user accounts to enable batch management activities.

When using POM and Customer Modules Management, it is required to enable or disable the application components based on implementation needs. The sections below describe which modules are needed to leverage the core integrations between applications on the platform. If a module is not present in the below lists, then it is only useful if you are implementing that

specific application and require that specific functionality. For example, most modules within Retail Insights are not required for the platform as a whole and can be disabled if you do not plan to implement RI.

After you make changes to the modules, make sure to synchronize your batch schedule in POM following the steps in Implementation Tools. If you are unsure whether a module should be enabled or not, you can initially leave it enabled and then disable jobs individually from POM as needed.

Common Modules

The following table lists the modules that are used across all platform implementations. These modules process core foundation data, generate important internal datasets, and move data downstream for other applications to use. Verify in your environment that these are visible in Customer Modules Management and are enabled.

Table 4-2 RAP Common Batch Modules

Application	Module	Usage Notes
RAP	COMMON > BATCH	Contains the minimum set of batch processes needed to load data into the platform and prepare it for one or more downstream applications (such as MFP or Science) when RI is not being implemented. This allows implementers to disable RI modules entirely to minimize the number of active batch jobs.
RAP	COMMON > RDXBATCH	Contains the batch processes for extracting data from RI to send to one or more Planning applications.
RAP	COMMON > ZIP_FILES	Choose which ZIP packages must be present before the nightly batch process begins. The batch only looks for the enabled files when starting and fails if any are not present.
RAP	COMMON > SIBATCH	Choose which input interfaces will be executed for loading flat files into RAP. Disabled interfaces will not look for or load the associated file even if it is provided.
RAP	COMMON > SICONTROLFILES	Choose which input files must be present before the nightly batch process begins. Active control files mark a file as required, and the batch fails if it is not provided. Inactive control files are treated as optional and are loaded if their associated job is enabled, but the batch will not fail if they are not provided.

RI Modules

The following table lists modules within the Retail Insights application which may be used by one or more RAP applications, in addition to the common modules from the prior section.

Table 4-3 RI Shared Batch Modules

Application	Module	Usage Notes
RI	COMMON	Use this section to choose your ZIP files for all flat file integrations. The same ZIP files can be selected from other application modules and they only need to be enabled in one place to become active.
RI	RSP > CONTROLFILES	Enable only the files you plan to send based on your RSP implementation plan. Refer to the *Interfaces Guide* for details. This is only used for non-foundation or legacy DAT files. This directly affects what files are required to be in the zip files, when an RI Daily Batch is executed. Required files that are missing will fail the batch.
RI	RSP > BATCH	Enable `RSP_REQUIRED` if you are implementing any Science module, plus others based on your application needs. For example, if you are loading sales then enable `RSP_SALES`. If you are loading inventory then enable `RSP_INVPOS`, and so on.

RSP Modules

The following table lists modules within the Science applications which may be used by one or more other RAP applications, in addition to the common modules from the prior section. This primarily covers forecasting and integration needs for Planning application usage. It is important to note that the underlying process for generating forecasts leverages jobs from the Promotion and Markdown Optimization (PMO) application, so you will see references to that product throughout the POM batch flows and in Retail Home modules.

Table 4-4 RSP Shared Batch Modules

Application	Module	Usage Notes
RSP	FCST > Batch	Collection of batch jobs associated with foundation loads, demand estimation, and forecast generation/export. The modules under this structure are needed to get forecasts for MFP.

To initialize data for the forecasting program, use the ad hoc POM process `RSE_MASTER_ADHOC_JOB` described in Sending Data to Science. After the platform is initialized, you may use the Forecast Configuration user interface to set up and run your initial forecasts. For complete details on the requirements and implementation process for forecasting, refer to the *Retail Science Platform Implementation Guide*.

For reference, the set of batch programs that should be enabled for forecasting are listed below (not including foundation data loads common to all of RSP). Enable these by syncing POM with MDF modules, though it is best to validate that the expected programs are enabled after the sync.

> **Note:**
>
> Jobs with PMO in the name are also used for Promotion & Markdown Optimization and are shared with the forecasting module.

Job Name
PMO_ACTIVITY_LOAD_START_JOB
PMO_ACTIVITY_STG_JOB
PMO_ACTIVITY_LOAD_JOB
PMO_ACTIVITY_LOAD_END_JOB
PMO_CREATE_BATCH_RUN_JOB
PMO_RUN_EXEC_SETUP_JOB
PMO_RUN_EXEC_START_JOB
PMO_RUN_EXEC_PROCESS_JOB
PMO_RUN_EXEC_END_JOB
RSE_CREATE_FCST_BATCH_RUN_JOB
RSE_FCST_BATCH_PROCESS_JOB
RSE_FCST_BATCH_RUN_END_JOB
RSE_CREATE_FCST_BATCH_RUN_ADHOC_JOB
RSE_FCST_BATCH_PROCESS_ADHOC_JOB

There are also two processes involved in forecast exports to MFP, one as part of weekly batch and the other as an ad hoc job which you can run during implementation.

Job Name
RSE_MFP_FCST_EXPORT_JOB
RSE_MFP_FCST_EXPORT_ADHOC_JOB

Adjustments in POM

While the bulk of your batch setup should be done in Retail Home, it may be necessary to fine-tune your schedule in POM after the initial configuration is complete. You may need to disable specific jobs in the nightly schedules (usually at Oracle's recommendation) or reconfigure the ad hoc processes to use different programs. The general steps to perform this activity are:

1. From Retail Home, click the link to navigate to POM or go to the POM URL directly if known. Log in as a batch administrator user.
2. Navigate to the Batch Administration screen.
3. Select the desired application tile, and then select the schedule type from the nightly, recurring, or standalone options.
4. Optionally, search for specific job names, and then use the **Enabled** option to turn the program on or off.

> **Note:**
>
> If you sync with MDF again in the future, it may re-enable jobs that you turned off inside a module that is turned on in Retail Home. For that reason, the module configuration is typically used only during implementation, then POM is used once you are live in production.

Self Service Features

The RI POM schedule includes jobs that support self-service actions where corrective action can be taken without waiting for assistance from Oracle. It is important to ensure the jobs below are enabled as part of your schedule, regardless of which platform modules you are using:

- `CTXFILE_VALIDATOR_JOB` – This job verifies the contents of your context files to ensure all non-nullable and required columns are being sent in the data files. The job will fail if it detects that any columns are missing, allowing you to take corrective action on the files you sent in the most recent upload.
- `RUN_PARTITION_VALIDATOR_JOB`– This job will verify the current state of your table partitions, check whether data is being loaded into the MAX partitions (which should not occur), and auto-create partitions if possible.
- `EXPSQLLDRLOGS_JOB` – This job will package and export any SQL Loader logs relating to invalid data which was rejected by the flat file load process and place it in an outgoing directory for download. The POM logs will generally show all summary information relating to these failures, but it can sometimes be useful to see the bad records and logs output by the SQL Loader script itself.

Additionally, there is an intraday process named `WEEKLY_MAINTENANCE_PROCESS` that can be scheduled from the RI POM standalone scheduler administration screen. This process should be scheduled to run daily (once enabled, it will automatically run most steps only weekly) and all jobs within it should be enabled. This process performs important upkeep activities such as database defragmentation, partition maintenance, and database statistics collection.

Configure POM Integrations

Retailers often have external applications that manage batch execution and automation of periodic processes. They might also require other downstream applications to be triggered automatically based on the status of programs in the Retail Analytics Platform. POM supports the following integrations that should be considered as part of your implementation plan.

Table 4-5 POM Integrations

Activity	References
Trigger RAP batches from an external program	*POM Implementation Guide* > Integration > Invoking Cycles in POM
Trigger external processes based on RAP batch statuses	*POM Implementation Guide* > Integration > External Status Update
Add external dependencies into the RAP batch to pause execution at specific points	*POM Implementation Guide* > Integration > External Dependency

Schedule the Batches

Once you are ready to begin regularly scheduled batch processing, you must log in to POM to enable each batch cycle and provide a start time for the earliest one in the batch sequence. The general steps are:

1. Log into POM as an administrator user.
2. Access the Scheduler Administration screen.
3. Select each available tile representing a batch schedule and select the **Nightly** batch option.
4. Edit the rows in the table to enable each batch and enter a start time. You only need to set a start time for the first batch in a linked sequence. For example, if you enter a start time for Retail Insights, then the RSP and MFP schedules will trigger automatically at the appropriate times, based on their dependencies and batch links.
5. For each batch after RI that must execute, you must also verify the inter-schedule dependencies are enabled for those batches. You can find the dependencies and enable them by going to the Batch Monitoring screen and selecting each batch's Nightly set of jobs. Click the number in front of Inter-Schedule Dependencies, and click **Enable** to the right of each displayed row if the current status is Disabled. This should change the status of the row to Pending.

Refer to the *POM User Guide* for additional information about the Scheduler screens and functionality. Scheduling the batch to run automatically is not required if you are using an external process or scheduler to trigger the batch instead.

For the Retail Insights nightly batch (which consumes the foundation input files for platform data), the batch will first look for all the required input ZIP packages configured through Customer Modules Management. The batch will wait for 4 hours for all required files, after which it will fail with an error in POM. When choosing a start time for the batch, it is best to start the 4-hour window an hour before you expect to have all the files uploaded for it to use. The batch performs reclassification of aggregate tables as its first step, which can take the full hour to complete in the case of very large reclassifications. This still provides 3+ hours during which you can schedule all file uploads and non-Oracle integration processes to occur.

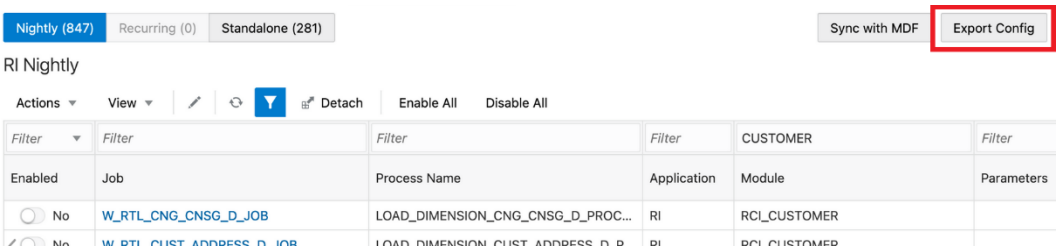
The other application batches, such as those for the Science modules, begin automatically once the required Retail Insights batch jobs are complete (assuming you have enabled those batches). Batch dependencies have been pre-configured to ensure each downstream process begins as soon as all required data is present.

Batch Flow Details

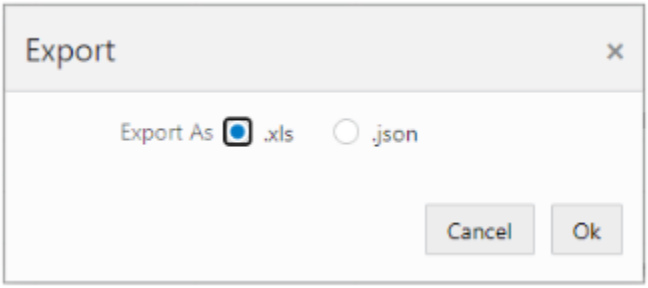
POM allows you to export the full batch schedule configuration to a spreadsheet for review. This is an easier way to see all of the jobs, processes, and dependencies that exist across all batch schedules that you are working with on the platform. Perform the following steps to access batch configuration details:

1. From Retail Home, click the link to navigate to POM, or go to the POM URL directly if known. Log in as a batch administrator user.
2. Navigate to the Batch Administration screen.

3. Select the desired application tile, and then select the schedule type from the nightly, recurring, or standalone options.
4. Click the **Export Config** button to download the schedule data.



5. Choose your preferred format (XLS or JSON).
   - XLS is meant for reviewing it visually.
   - JSON is used for importing the schedule to another environment.
6. Save the file to disk and open it to review the contents.



For more details about the tabs in the resulting XLS file, refer to the *POM User Guide*, “Export/Import Schedule Configuration”.

Planning Applications Job Details

Planning applications such as Merchandise Financial Planning schedule jobs through POM, and run ad hoc tasks directly using Online Administration Tasks (OAT) within the application. POM is the preferred approach for scheduling jobs in RAP for managing interdependencies with jobs from RI and Science. Refer to the application Administration Guides for more details on scheduling tasks through OAT.

Planning applications also use a special Batch Framework that controls jobs using batch control files. You may configure these batch control files for various supported changes within the application. The POM schedule for Planning internally calls an OAT task controlled by the batch control entries. A standard set of nightly and weekly jobs for Planning are defined to schedule them in POM. You also have the option to disable or enable the jobs either directly through POM, or by controlling the entries in batch control files. Refer to the Planning application-specific Implementation Guides for details about the list of jobs and how jobs can be controlled by making changes to batch control files.

5
Transformations

The Retail Analytics Platform is a data-driven set of applications performing many transformations and calculations as part of normal operations. Review this chapter to learn about the most common types of data transformation activities occurring within the platform that may impact the data you send into the platform and the results you see as an end user.

Aggregate Tables in RI

Retail Insights accepts most data at a common base intersection of item, location, and date. However, downstream applications such as Planning or Science may need this data at higher levels of aggregation than that. To support those data needs, RI pre-aggregates the incoming fact data to certain higher levels, depending on the requirements of the consuming application. For Planning purposes, all fact data is aggregated to a common level of item, location, and fiscal week before it is exported for downstream consumption. Review the sections below to learn more about how data moves through Retail Insights for other applications to use.

Table Structures

If you are currently loading data into RI and need to access database tables for debugging or validation purposes, there are naming and format conventions used on each aggregate table. A base intersection table is abbreviated using the following notations:

Table 5-1 RI Base Fact Structure

Table Name Component	Explanation
W_	Most RI tables start with a “W” to denote a core data warehouse table.
IT	Abbreviation for Item
LC	Abbreviation for Location
DY	Abbreviation for Day
WK	Abbreviation for Week
_D	Dimensional tables end with “D”
_F	Base intersection fact tables end with “F”
_A	Aggregate tables end with “A”

Using the above notation, you may interpret the table `W_RTL_SLS_IT_LC_WK_A` as “Sales aggregate table at the item/location/week intersection”.

Key Columns

Most fact tables in RI use the same key column structure, which consists of two types of internal identifiers. The first identifier is referred to as a WID value. The WID on a fact table is

a foreign key reference to a dimension table's ROW_WID column. For example, a PROD_WID column in a sales table is referring to the ROW_WID on W_PRODUCT_D (the product dimension table). Joining the WIDs on a fact and a dimension will allow you to look up user-facing descriptors for the dimensions, such as the product number.

The second identifier is known as SCD1_WID and refers to slowly changing dimensions, which is a common data warehousing concept. The IDs on the SCD1_WID columns are carried forward through reclassifications and other dimensional changes, allowing you to locate a single product throughout history, even if it has numerous records in the parent dimension table. For example, joining PROD_SCD1_WID from a sales table with SCD1_WID on W_PRODUCT_D will receive all instances of that product's data throughout history, even if the product has several different ROW_WID entries due to reclassifications, which insert new records to the dimension for the same item.

The other core structure to understand is Date WIDs (key column DT_WID). These also join with the ROW_WID of the parent dimension (W_MCAL_DAY_D usually), but the format of the WID allows you to extract the date value directly if needed, without table joins. The standard DT_WID value used is a combination of 1 + date in YYYYMMDD + 000. For example, 120210815000 is the DT_WID value for "August 15, 2021".

Transformations from RI to Planning

Data Filtering and Conversions

In addition to simply aggregating the incoming fact data from item/location/date to item/location/week level, it is also important to understand what RI is doing with the data as it moves from the input files to the outbound interfaces. The table below summarizes the transformations and business logic applied to shared facts used by RAP applications.

Table 5-2 RI Shared Data Transformations

Transformation	Explanation
Currency Conversion	As part of the nightly batch, RI will use exchange rate information to convert all incoming data from the source currency to the primary business currency. All data sent to downstream applications is in the primary currency. The RI data model maintains separate columns for both local and primary currency amounts for BI usage.
Tax Handling	RI includes non-US taxes, such as VAT, in the sales retail amounts based on the indicators set up in the source system (such as Sales Audit). When sending the sales data to Planning and Science, the default sales values may include VAT and only specific VAT-exclusive fields will remove it.
Transaction Date Usage	All fact data coming into the system includes a transaction date on the record. RI aggregates from day to week level using transaction dates and does not alter or re-assign any records to different dates from what is provided. Transaction data in the past will be added to their historical week in the aggregates, no matter how far back it is dated.

Table 5-2 (Cont.) RI Shared Data Transformations

Transformation	Explanation
Pack Item Handling	Downstream applications are currently only interested in the component item level, so RI will not send any data for pack items to other applications. Pack item sales must be spread to the component item level and loaded into the Sales Pack interface if this data is required for Science or Planning. All inventory, purchase order, and transaction data must be loaded at the component item level only.
Stockholding Locations	Inventory data for Planning is only exported for stockholding locations. A store indicated as a non-stockholding location on the location dimension will not be included in outbound data. Physical warehouses which are not stockholding (because you use virtual warehouses) will also not be included.
Future On Order	Planning applications require a forward-looking view of purchase orders based on the OTB EOW Date. RI accepts the actual purchase order details on the interfaces but will then transform the on-order amounts to be future-dated using the provided OTB EOW Dates. Orders which are past the OTB date will be included in the first EOW date, they will never be in the past.
Include On Order	Purchase Order data is limited by the Include On Order Flag on the Order Head interface. A value of `N` will not be included in the calculations for Planning.
Orderable Items	Purchase Order data is limited by the Orderable Flag on the Product interface. A value of `N` will not be included in the calculations for Planning.
Inventory Adjustment Types	RI accepts 3 types of inventory adjustments using the codes `22`, `23`, and `41`. For Planning, only the first two codes are exported. Code `22` relates to Shrink and code `23` relates to Non-Shrink.
Inventory Receipt Types	RI accepts 3 types of inventory receipts using the codes `20`, `44~T`, and `44~A`. For Planning, all codes are sent but the `44`s are summed together. Code `20` relates to purchase order receipts. Code `44` relates to Transfer receipts and Allocation receipts.
Inventory Transfer Types	RI accepts 3 types of transfers using the codes `N`, `B`, and `I` (normal, book, and intercompany). All three types are sent to planning along with the type codes.

Data Mappings

When you are generating input files to RAP, you may also want to know which columns are being moved to the output and how that data translates from what you see in the file to what you see in Planning applications. The list of mappings below describes how the data in the Retail Insights data model is exported to PDS.

> **Note:**
>
> Conversions and filters listed in the prior section of this chapter apply to all of this data (for example, data may be stored in local currency in RI but is always converted to the primary currency for export).

Sales Mapping

Data for sales is loaded from the `SALES.csv` file or from RMFCS (Sales Audit). The primary RI table is the week-level aggregate generated by the historical load process. All data mappings in this area are split out by retail type. Any measure having `reg/pro/clr` in the name are being filtered on that retail type code as part of the export. When you provide input data to RAP, you specify the retail type code as `R`, `P`, or `C`, and those values are used here to determine the output.

Measure	Target Table	Target Column	RI Data Source
Gross Reg Sales Units	W_PDS_SLS_IT_LC_WK_A	SALES_REG_UNITS	W_RTL_SLS_IT_LC_WK_A.SLS_QTY + W_RTL_SLSPK_IT_LC_WK_A.SLSPK_QTY
Gross Pro Sales Units	W_PDS_SLS_IT_LC_WK_A	SALES_PRO_UNITS	W_RTL_SLS_IT_LC_WK_A.SLS_QTY + W_RTL_SLSPK_IT_LC_WK_A.SLSPK_QTY
Gross Clr Sales Units	W_PDS_SLS_IT_LC_WK_A	SALES_CLR_UNITS	W_RTL_SLS_IT_LC_WK_A.SLS_QTY + W_RTL_SLSPK_IT_LC_WK_A.SLSPK_QTY
Gross Reg Sales Cost	W_PDS_SLS_IT_LC_WK_A	SALES_REG_COST	(W_RTL_SLS_IT_LC_WK_A.SLS_AMT-SLS_PROFIT_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT-SLSPK_PROF_AMT)
Gross Pro Sales Cost	W_PDS_SLS_IT_LC_WK_A	SALES_PRO_COST	(W_RTL_SLS_IT_LC_WK_A.SLS_AMT-SLS_PROFIT_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT-SLSPK_PROF_AMT)
Gross Clr Sales Cost	W_PDS_SLS_IT_LC_WK_A	SALES_CLR_COST	(W_RTL_SLS_IT_LC_WK_A.SLS_AMT-SLS_PROFIT_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT-SLSPK_PROF_AMT)
Gross Reg Sales Retail	W_PDS_SLS_IT_LC_WK_A	SALES_REG_RETAIL	W_RTL_SLS_IT_LC_WK_A.SLS_AMT + W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT
Gross Pro Sales Retail	W_PDS_SLS_IT_LC_WK_A	SALES_PRO_RETAIL	W_RTL_SLS_IT_LC_WK_A.SLS_AMT + W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT
Gross Clr Sales Retail	W_PDS_SLS_IT_LC_WK_A	SALES_CLR_RETAIL	W_RTL_SLS_IT_LC_WK_A.SLS_AMT + W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT

Measure	Target Table	Target Column	RI Data Source
Gross Reg Sales Tax	W_PDS_SLS_IT_LC_WK_A	SALES_REG_TAX	W_RTL_SLS_IT_LC_WK_A.SLS_TAX_AMT + W_RTL_SLSPK_IT_LC_WK_A.SLSPK_TAX_AMT
Gross Pro Sales Tax	W_PDS_SLS_IT_LC_WK_A	SALES_PRO_TAX	W_RTL_SLS_IT_LC_WK_A.SLS_TAX_AMT + W_RTL_SLSPK_IT_LC_WK_A.SLSPK_TAX_AMT
Gross Clr Sales Tax	W_PDS_SLS_IT_LC_WK_A	SALES_CLR_TAX	W_RTL_SLS_IT_LC_WK_A.SLS_TAX_AMT + W_RTL_SLSPK_IT_LC_WK_A.SLSPK_TAX_AMT
Net Reg Sales Units	W_PDS_SLS_IT_LC_WK_A	NET_SALES_REG_UNITS	(W_RTL_SLS_IT_LC_WK_A.SLS_QTY-RET_QTY) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_QTY-RETPK_QTY)
Net Pro Sales Units	W_PDS_SLS_IT_LC_WK_A	NET_SALES_PRO_UNITS	(W_RTL_SLS_IT_LC_WK_A.SLS_QTY-RET_QTY) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_QTY-RETPK_QTY)
Net Clr Sales Units	W_PDS_SLS_IT_LC_WK_A	NET_SALES_CLR_UNITS	(W_RTL_SLS_IT_LC_WK_A.SLS_QTY-RET_QTY) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_QTY-RETPK_QTY)
Net Reg Sales Cost	W_PDS_SLS_IT_LC_WK_A	NET_SALES_REG_COST	(W_RTL_SLS_IT_LC_WK_A.SLS_AMT-RET_AMT) - (W_RTL_SLS_IT_LC_WK_A.SLS_PROFIT_AMT-RET_PROFIT_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT-RETPK_AMT) - (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_PROF_AMT-RETPK_PROF_AMT)
Net Pro Sales Cost	W_PDS_SLS_IT_LC_WK_A	NET_SALES_PRO_COST	(W_RTL_SLS_IT_LC_WK_A.SLS_AMT-RET_AMT) - (W_RTL_SLS_IT_LC_WK_A.SLS_PROFIT_AMT-RET_PROFIT_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT-RETPK_AMT) - (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_PROF_AMT-RETPK_PROF_AMT)
Net Clr Sales Cost	W_PDS_SLS_IT_LC_WK_A	NET_SALES_CLR_COST	(W_RTL_SLS_IT_LC_WK_A.SLS_AMT-RET_AMT) - (W_RTL_SLS_IT_LC_WK_A.SLS_PROFIT_AMT-RET_PROFIT_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT-RETPK_AMT) - (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_PROF_AMT-RETPK_PROF_AMT)

Measure	Target Table	Target Column	RI Data Source
Net Reg Sales Retail	W_PDS_SLS_IT_LC_WK_A	NET_SALES_REG_RETAIL	(W_RTL_SLS_IT_LC_WK_A.SLS_AMT-RET_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT-RETPK_AMT)
Net Pro Sales Retail	W_PDS_SLS_IT_LC_WK_A	NET_SALES_PRO_RETAIL	(W_RTL_SLS_IT_LC_WK_A.SLS_AMT-RET_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT-RETPK_AMT)
Net Clr Sales Retail	W_PDS_SLS_IT_LC_WK_A	NET_SALES_CLR_RETAIL	(W_RTL_SLS_IT_LC_WK_A.SLS_AMT-RET_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT-RETPK_AMT)
Net Reg Sales Retail Excluding VAT	W_PDS_SLS_IT_LC_WK_A	NET_SALES_REG_RETAIL_VAT_EXCL	(W_RTL_SLS_IT_LC_WK_A.SLS_AMT-RET_AMT) - (W_RTL_SLS_IT_LC_WK_A.SLS_TAX_AMT-RET_TAX_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT-RETPK_AMT) - (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_TAX_AMT-RETPK_TAX_AMT)
Net Pro Sales Retail Excluding VAT	W_PDS_SLS_IT_LC_WK_A	NET_SALES_PRO_RETAIL_VAT_EXCL	(W_RTL_SLS_IT_LC_WK_A.SLS_AMT-RET_AMT) - (W_RTL_SLS_IT_LC_WK_A.SLS_TAX_AMT-RET_TAX_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT-RETPK_AMT) - (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_TAX_AMT-RETPK_TAX_AMT)
Net Clr Sales Retail Excluding VAT	W_PDS_SLS_IT_LC_WK_A	NET_SALES_CLR_RETAIL_VAT_EXCL	(W_RTL_SLS_IT_LC_WK_A.SLS_AMT-RET_AMT) - (W_RTL_SLS_IT_LC_WK_A.SLS_TAX_AMT-RET_TAX_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_AMT-RETPK_AMT) - (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_TAX_AMT-RETPK_TAX_AMT)
Net Reg Sales Tax	W_PDS_SLS_IT_LC_WK_A	NET_SALES_REG_TAX	(W_RTL_SLS_IT_LC_WK_A.SLS_TAX_AMT-RET_TAX_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_TAX_AMT-RETPK_TAX_AMT)
Net Pro Sales Tax	W_PDS_SLS_IT_LC_WK_A	NET_SALES_PRO_TAX	(W_RTL_SLS_IT_LC_WK_A.SLS_TAX_AMT-RET_TAX_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_TAX_AMT-RETPK_TAX_AMT)
Net Clr Sales Tax	W_PDS_SLS_IT_LC_WK_A	NET_SALES_CLR_TAX	(W_RTL_SLS_IT_LC_WK_A.SLS_TAX_AMT-RET_TAX_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.SLSPK_TAX_AMT-RETPK_TAX_AMT)

Measure	Target Table	Target Column	RI Data Source
Returns Reg Units	W_PDS_SLS_IT_LC_WK_A	RETURNS_REG_UNITS	W_RTL_SLS_IT_LC_WK_A.RET_QTY + W_RTL_SLSPK_IT_LC_WK_A.RETPK_QTY
Returns Pro Units	W_PDS_SLS_IT_LC_WK_A	RETURNS_PRO_UNITS	W_RTL_SLS_IT_LC_WK_A.RET_QTY + W_RTL_SLSPK_IT_LC_WK_A.RETPK_QTY
Returns Clr Units	W_PDS_SLS_IT_LC_WK_A	RETURNS_CLR_UNITS	W_RTL_SLS_IT_LC_WK_A.RET_QTY + W_RTL_SLSPK_IT_LC_WK_A.RETPK_QTY
Returns Reg Cost	W_PDS_SLS_IT_LC_WK_A	RETURNS_REG_COST	(W_RTL_SLS_IT_LC_WK_A.RET_AMT-RET_PROFIT_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.RETPK_AMT-RETPK_PROF_AMT)
Returns Pro Cost	W_PDS_SLS_IT_LC_WK_A	RETURNS_PRO_COST	(W_RTL_SLS_IT_LC_WK_A.RET_AMT-RET_PROFIT_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.RETPK_AMT-RETPK_PROF_AMT)
Returns Clr Cost	W_PDS_SLS_IT_LC_WK_A	RETURNS_CLR_COST	(W_RTL_SLS_IT_LC_WK_A.RET_AMT-RET_PROFIT_AMT) + (W_RTL_SLSPK_IT_LC_WK_A.RETPK_AMT-RETPK_PROF_AMT)
Returns Reg Retail	W_PDS_SLS_IT_LC_WK_A	RETURNS_REG_RETAIL	W_RTL_SLS_IT_LC_WK_A.RET_AMT + W_RTL_SLSPK_IT_LC_WK_A.RETPK_AMT
Returns Pro Retail	W_PDS_SLS_IT_LC_WK_A	RETURNS_PRO_RETAIL	W_RTL_SLS_IT_LC_WK_A.RET_AMT + W_RTL_SLSPK_IT_LC_WK_A.RETPK_AMT
Returns Clr Retail	W_PDS_SLS_IT_LC_WK_A	RETURNS_CLR_RETAIL	W_RTL_SLS_IT_LC_WK_A.RET_AMT + W_RTL_SLSPK_IT_LC_WK_A.RETPK_AMT
Returns Reg Tax	W_PDS_SLS_IT_LC_WK_A	RETURNS_REG_TAX	W_RTL_SLS_IT_LC_WK_A.RET_TAX_AMT + W_RTL_SLSPK_IT_LC_WK_A.RETPK_TAX_AMT
Returns Pro Tax	W_PDS_SLS_IT_LC_WK_A	RETURNS_PRO_TAX	W_RTL_SLS_IT_LC_WK_A.RET_TAX_AMT + W_RTL_SLSPK_IT_LC_WK_A.RETPK_TAX_AMT
Returns Clr Tax	W_PDS_SLS_IT_LC_WK_A	RETURNS_CLR_TAX	W_RTL_SLS_IT_LC_WK_A.RET_TAX_AMT + W_RTL_SLSPK_IT_LC_WK_A.RETPK_TAX_AMT

Inventory Position Mapping

Data is loaded from the `INVENTORY.csv` file or from RMFCS. The primary RI table is the week-level aggregate generated by the historical load process.

Measure	Target Table	Target Column	RI Data Source
Inventory Units	W_PDS_INV_IT_LC_WK_A	REGULAR_INVENTORY_UNITS	W_RTL_INV_IT_LC_WK_A.INV_SOH_QTY + INV_IN_TRAN_QTY
Inventory Cost	W_PDS_INV_IT_LC_WK_A	REGULAR_INVENTORY_COST	W_RTL_INV_IT_LC_WK_A.INV_SOH_COST_AMT + INV_IN_TRAN_COST_AMT
Inventory Retail	W_PDS_INV_IT_LC_WK_A	REGULAR_INVENTORY_RETAIL	W_RTL_INV_IT_LC_WK_A.INV_SOH_RTL_AMT + INV_IN_TRAN_RTL_AMT
Unit Cost	W_PDS_INV_IT_LC_WK_A	UNIT_COST	W_RTL_INV_IT_LC_WK_A.INV_UNIT_COST_AMT
Average Cost	W_PDS_INV_IT_LC_WK_A	AV_COST	W_RTL_INV_IT_LC_WK_A.INV_AVG_COST_AMT
Unit Retail	W_PDS_INV_IT_LC_WK_A	UNIT_RETAIL	W_RTL_INV_IT_LC_WK_A.INV_UNIT_RTL_AMT

On Order Mapping

Data is loaded from the `ORDER_HEAD.csv` and `ORDER_DETAIL.csv` files or from RMFCS. Purchase order data is transformed from the raw order line details into a forward-looking total on-order amount based on the OTB end-of-week date on the order. Data is also filtered to remove orders not flagged as Include On Order.

Measure	Target Table	Target Column	RI Data Source
On Order Units	W_PDS_PO_ONORD_IT_LC_WK_A	ON_ORDER_UNITS	W_RTL_PO_ONORD_IT_LC_DY_F.PO_ONORD_QTY
On Order Cost	W_PDS_PO_ONORD_IT_LC_WK_A	ON_ORDER_COST	W_RTL_PO_ONORD_IT_LC_DY_F.PO_ONORD_COST_AMT_LCL
On Order Retail	W_PDS_PO_ONORD_IT_LC_WK_A	ON_ORDER_RETAIL	W_RTL_PO_ONORD_IT_LC_DY_F.PO_ONORD_RTL_AMT_LCL

Markdown Mapping

Data is loaded from the `MARKDOWN.csv` file or from RMFCS. The primary RI table is the week-level aggregate generated by the historical load process.

Measure	Target Table	Target Column	RI Data Source
Regular Markdown	W_PDS_MKDN_IT_LC_WK_A	REG_MARKDOWN_RETAIL	W_RTL_MKDN_IT_LC_WK_A.MKDN_AMT where retail type = R

Measure	Target Table	Target Column	RI Data Source
Promotion Markdown (Regular)	W_PDS_MKDN_IT_LC_WK_A	PROMO_MARKDOWN_RETAIL_REG	W_RTL_MKDN_IT_LC_WK_A.MKDN_AMT where retail type = P and CLEARANCE_FLG = N
Promotion Markdown (Clearance)	W_PDS_MKDN_IT_LC_WK_A	PROMO_MARKDOWN_RETAIL_CLEAR	W_RTL_MKDN_IT_LC_WK_A.MKDN_AMT where retail type = P and CLEARANCE_FLG = Y
Clearance Markdown	W_PDS_MKDN_IT_LC_WK_A	CLEAR_MARKDOWN_RETAIL	W_RTL_MKDN_IT_LC_WK_A.MKDN_AMT where retail type = C
Markup	W_PDS_MKDN_IT_LC_WK_A	MARKUP	W_RTL_MKDN_IT_LC_WK_A.MKUP_AMT where retail type = R
Markup Cancel	W_PDS_MKDN_IT_LC_WK_A	MARKUP_CANCEL	W_RTL_MKDN_IT_LC_WK_A.MKUP_CAN_AMT
Markdown Cancel	W_PDS_MKDN_IT_LC_WK_A	MARKDOWN_CANCEL	W_RTL_MKDN_IT_LC_WK_A.MKDN_CAN_AMT
Intercompany Markup	W_PDS_MKDN_IT_LC_WK_A	INTERCOMPANY_MARKUP	W_RTL_MKDN_IT_LC_WK_A.MKUP_AMT where retail type = I
Intercompany Markdown	W_PDS_MKDN_IT_LC_WK_A	INTERCOMPANY_MARKDOWN	W_RTL_MKDN_IT_LC_WK_A.MKDN_AMT where retail type = I

Wholesale/Franchise Mapping

Data is loaded from the `SALES_WF.csv` file or from RMFCS. The primary RI table is the week-level aggregate generated by the historical load process.

Measure	Target Table	Target Column	RI Data Source
Franchise Sales Units	W_PDS_SLSWF_IT_LC_WK_A	FRANCHISE_SALES_UNITS	W_RTL_SLSWF_IT_LC_WK_A.SLSWF_QTY
Franchise Sales Cost	W_PDS_SLSWF_IT_LC_WK_A	FRANCHISE_SALES_COST	W_RTL_SLSWF_IT_LC_WK_A.SLSWF_ACQ_COST_AMT
Franchise Sales Retail	W_PDS_SLSWF_IT_LC_WK_A	FRANCHISE_SALES_RETAIL	W_RTL_SLSWF_IT_LC_WK_A.SLSWF_AMT
Franchise Sales Tax	W_PDS_SLSWF_IT_LC_WK_A	FRANCHISE_SALES_TAX	W_RTL_SLSWF_IT_LC_WK_A.SLSWF_TAX_AMT
Franchise Returns Units	W_PDS_SLSWF_IT_LC_WK_A	FRANCHISE_RETURNS_UNITS	W_RTL_SLSWF_IT_LC_WK_A.RETWF_QTY
Franchise Returns Cost	W_PDS_SLSWF_IT_LC_WK_A	FRANCHISE_RETURNS_COST	W_RTL_SLSWF_IT_LC_WK_A.RETWF_ACQ_COST_AMT
Franchise Returns Retail	W_PDS_SLSWF_IT_LC_WK_A	FRANCHISE_RETURNS_RETAIL	W_RTL_SLSWF_IT_LC_WK_A.RETWF_AMT
Franchise Returns Tax	W_PDS_SLSWF_IT_LC_WK_A	FRANCHISE_RETURNS_TAX	W_RTL_SLSWF_IT_LC_WK_A.RETWF_TAX_AMT

Measure	Target Table	Target Column	RI Data Source
Franchise Restocking Fee	W_PDS_SLSWF_IT_LC_WK_A	FRANCHISE_RESTOCK_FEE	W_RTL_SLSWF_IT_LC_WK_A.RETWF_RSTK_FEE_AMT
Franchise Markdown	W_PDS_SLSWF_IT_LC_WK_A	WF_MARKDOWN_RETAIL	W_RTL_SLSWF_IT_LC_WK_A.SLSWF_MKDN_AMT - RETWF_MKDN_AMT
Franchise Markup	W_PDS_SLSWF_IT_LC_WK_A	WF_MARKUP_RETAIL	W_RTL_SLSWF_IT_LC_WK_A.SLSWF_MKUP_AMT - RETWF_MKUP_AMT

Inventory Adjustments Mapping

Data is loaded from the `ADJUSTMENT.csv` file or from RMFCS. The primary RI table is the week-level aggregate generated by the historical load process.

Measure	Target Table	Target Column	RI Data Source
Shrink Units	W_PDS_INVADJ_IT_LC_WK_A	SHRINK_UNITS	W_RTL_INVADJ_IT_LC_WK_A.INVADJ_QTY where adj type = 22
Shrink Cost	W_PDS_INVADJ_IT_LC_WK_A	SHRINK_COST	W_RTL_INVADJ_IT_LC_WK_A.INVADJ_COST_AMT where adj type = 22
Shrink Retail	W_PDS_INVADJ_IT_LC_WK_A	SHRINK_RETAIL	W_RTL_INVADJ_IT_LC_WK_A.INVADJ_RTL_AMT where adj type = 22
Non-Shrink Adjustments Units	W_PDS_INVADJ_IT_LC_WK_A	NON_SHRINK_ADJ_UNITS	W_RTL_INVADJ_IT_LC_WK_A.INVADJ_QTY where adj type = 23
Non-Shrink Adjustments Cost	W_PDS_INVADJ_IT_LC_WK_A	NON_SHRINK_ADJ_COST	W_RTL_INVADJ_IT_LC_WK_A.INVADJ_COST_AMT where adj type = 23
Non-Shrink Adjustments Retail	W_PDS_INVADJ_IT_LC_WK_A	NON_SHRINK_ADJ_RETAIL	W_RTL_INVADJ_IT_LC_WK_A.INVADJ_RTL_AMT where adj type = 23

Inventory Receipts Mapping

Data is loaded from the `RECEIPT.csv` file or from RMFCS. The primary RI table is the week-level aggregate generated by the historical load process.

Measure	Target Table	Target Column	RI Data Source
PO Receipt Units	W_PDS_INVRC_IT_LC_WK_A	PO_RECEIPT_UNITS	W_RTL_INVRC_IT_LC_WK_A.INVRC_QTY where rcpt type code = 20
PO Receipt Cost	W_PDS_INVRC_IT_LC_WK_A	PO_RECEIPT_COST	W_RTL_INVRC_IT_LC_WK_A.INVRC_COST_AMT where rcpt type code = 20
PO Receipt Retail	W_PDS_INVRC_IT_LC_WK_A	PO_RECEIPT_RETAIL	W_RTL_INVRC_IT_LC_WK_A.INVRC_RTL_AMT where rcpt type code = 20
Transfer/ Allocation Receipt Units	W_PDS_INVRC_IT_LC_WK_A	TSF_RECEIPT_UNITS	W_RTL_INVRC_IT_LC_WK_A.INVRC_QTY where rcpt type code = 44~A or 44~T

Measure	Target Table	Target Column	RI Data Source
Transfer/ Allocation Receipt Cost	W_PDS_INVRC_IT_LC_WK_A	TSF_RECEIPT_COST	W_RTL_INVRC_IT_LC_WK_A.INVRC_COST_AMT where rcpt type code = 44~A or 44~T
Transfer/ Allocation Receipt Retail	W_PDS_INVRC_IT_LC_WK_A	TSF_RECEIPT_RETAIL	W_RTL_INVRC_IT_LC_WK_A.INVRC_RTL_AMT where rcpt type code = 44~A or 44~T

Inventory Transfers Mapping

Data is loaded from the `TRANSFER.csv` file or from RMFCS. The primary RI table is the week-level aggregate generated by the historical load process.

Measure	Target Table	Target Column	RI Data Source
Transfer Type	W_PDS_INVTSF_IT_LC_WK_A	TSF_TYPE	W_XACT_TYPE_D.W_XACT_TYPE_CODE in (N,B,I) (for normal/book/intercompany tsfs)
Transfer Inbound Units	W_PDS_INVTSF_IT_LC_WK_A	TSF_IN_UNITS	W_RTL_INVTSF_IT_LC_WK_A.TSF_TO_LOC_QTY
Transfer Inbound Cost	W_PDS_INVTSF_IT_LC_WK_A	TSF_IN_COST	W_RTL_INVTSF_IT_LC_WK_A.TSF_TO_LOC_COST_AMT
Transfer Inbound Retail	W_PDS_INVTSF_IT_LC_WK_A	TSF_IN_RETAIL	W_RTL_INVTSF_IT_LC_WK_A.TSF_TO_LOC_RTL_AMT
Transfer Outbound Units	W_PDS_INVTSF_IT_LC_WK_A	TSF_OUT_UNITS	W_RTL_INVTSF_IT_LC_WK_A.TSF_FROM_LOC_QTY
Transfer Outbound Cost	W_PDS_INVTSF_IT_LC_WK_A	TSF_OUT_COST	W_RTL_INVTSF_IT_LC_WK_A.TSF_FROM_LOC_COST_AMT
Transfer Outbound Retail	W_PDS_INVTSF_IT_LC_WK_A	TSF_OUT_RETAIL	W_RTL_INVTSF_IT_LC_WK_A.TSF_FROM_LOC_RTL_AMT

Inventory RTVs Mapping

Data is loaded from the `RTV.csv` file or from RMFCS. The primary RI table is the week-level aggregate generated by the historical load process.

Measure	Target Table	Target Column	RI Data Source
RTV Units	W_PDS_INVRTV_IT_LC_WK_A	RTV_UNITS	W_RTL_INVRTV_IT_LC_WK_A.RTV_QTY
RTV Cost	W_PDS_INVRTV_IT_LC_WK_A	RTV_COST	W_RTL_INVRTV_IT_LC_WK_A.RTV_COST_AMT
RTV Retail	W_PDS_INVRTV_IT_LC_WK_A	RTV_RETAIL	W_RTL_INVRTV_IT_LC_WK_A.RTV_RTL_AMT

Measure	Target Table	Target Column	RI Data Source
RTV Restocking Fee	W_PDS_INVRTV_IT_LC_WK_A	RTV_RESTOCK_FEE	W_RTL_INVRTV_IT_LC_WK_A.RTV_RSTCK_COST_AMT

Inventory Reclass Mapping

Data is loaded from the `INV_RECLASS.csv` file or from RMFCS. The primary RI table is the week-level aggregate generated by the historical load process.

Measure	Target Table	Target Column	RI Data Source
Reclass In Units	W_PDS_INVRECLASS_IT_LC_WK_A	RECLASS_IN_UNITS	W_RTL_INVRCL_IT_LC_WK_A.RCL_TO_LOC_QTY
Reclass In Cost	W_PDS_INVRECLASS_IT_LC_WK_A	RECLASS_IN_COST	W_RTL_INVRCL_IT_LC_WK_A.RCL_TO_LOC_COST_AMT
Reclass In Retail	W_PDS_INVRECLASS_IT_LC_WK_A	RECLASS_IN_RETAIL	W_RTL_INVRCL_IT_LC_WK_A.RCL_TO_LOC_RTL_AMT
Reclass Out Units	W_PDS_INVRECLASS_IT_LC_WK_A	RECLASS_OUT_UNITS	W_RTL_INVRCL_IT_LC_WK_A.RCL_FROM_LOC_QTY
Reclass Out Cost	W_PDS_INVRECLASS_IT_LC_WK_A	RECLASS_OUT_COST	W_RTL_INVRCL_IT_LC_WK_A.RCL_FROM_LOC_COST_AMT
Reclass Out Retail	W_PDS_INVRECLASS_IT_LC_WK_A	RECLASS_OUT_RETAIL	W_RTL_INVRCL_IT_LC_WK_A.RCL_FROM_LOC_RTL_AMT

Deal Income Mapping

Data is loaded from the `DEAL_INCOME.csv` file or from RMFCS. The primary RI table is the week-level aggregate generated by the historical load process.

Measure	Target Table	Target Column	RI Data Source
Deal Income Sales Based	W_PDS_DEALINC_IT_LC_WK_A	DEAL_INCOME_SALES	W_RTL_DEALINC_IT_LC_WK_A.DEAL_SLS_COST_AMT
Deal Income Purchases Based	W_PDS_DEALINC_IT_LC_WK_A	DEAL_INCOME_PURCHASES	W_RTL_DEALINC_IT_LC_WK_A.DEAL_PRCH_COST_AMT

Intercompany Margin Mapping

Data is loaded from the `IC_MARGIN.csv` file or from RMFCS. The primary RI table is the week-level aggregate generated by the historical load process.

Measure	Target Table	Target Column	RI Data Source
Intercompany Margin	W_PDS_ICM_IT_LC_WK_A	INTERCOMPANY_MARGIN	W_RTL_ICM_IT_LC_WK_A.IC_MARGIN_AMT

Transformations in Planning

Planning applications allow the loading of fact data at the load intersection level (such as Item and Location) but uses the data within the application at an aggregated level (called the base intersection). In MFP, though all facts are loaded at the item level, it only needs data to plan at the Subclass level. The data will be aggregated from item level to subclass level for all the configured metrics to be directly used by the application. During re-classifications (such as when one item is moved from one subclass to another subclass), after the new hierarchy details are imported into MFP it also triggers re-classification of all fact data. Re-aggregation of fact data then happens only for shared facts having different load and base intersections.

In Planning applications, fact data is grouped as dynamic fact groups based mainly on the base intersection and interface details, as defined in the Data Interface of the Application Configuration. RI and Science use a relational data model, whereas Planning applications internally use a hierarchical data model. Data from RAP, stored using the relational data model, needs to be transformed to be loaded into Planning applications. A similar approach is necessary for data coming out of planning applications to Science or external sources. These data transformations happen as part of the interfaces defined in `interface.cfg` (Interfaces Configuration File), which is a mapping of dimensions and measures from Planning applications to external system table columns. Refer to the application-specific Implementation Guides for more information about Planning Data Interfaces.

6
Implementation Tools

Review the sections below to learn about the tools and common components used within the Retail Analytics Platform. Many of these tools are used both for initial implementation and for ongoing maintenance activities, so implementers should be prepared to transfer knowledge of these tools to the customer before completing the project.

Retail Home

One of the first places you will go in a new RAP environment is Retail Home. It serves both as the customer portal for Oracle Retail cloud applications and as a centralized place for certain common configurations, such as Customer Module Management. Module management allows implementers to quickly configure the complex batch schedules and interdependencies of RAP applications using a simplified module-based layout. Optional batch programs, such as those used for Retail Insights or Science applications, can be turned off from this tool and it synchronizes with the batch scheduler to ensure all related programs are disabled automatically.

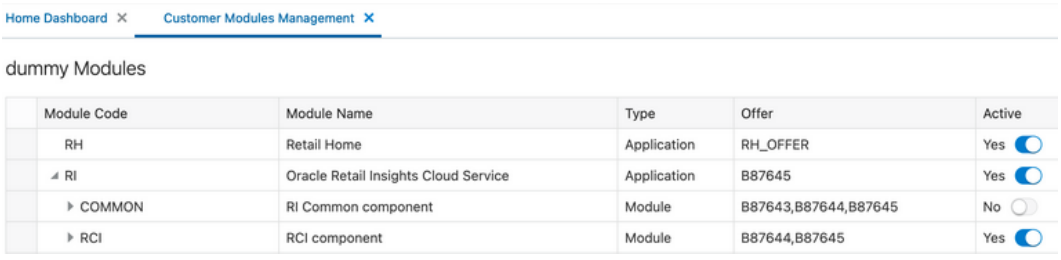
For more general information about Retail Home and the other features it provides, review the *Retail Home Administration Guide*.

Because Customer Modules are a necessary part of configuring and using a RAP environment, see the steps below for how to access this feature.

1. To access Retail Home, access the URL sent to your cloud administrator on first provisioning a new environment. It should look similar to the URL format below.

   ```
   https://{service}.retail.{region}.ocs.oraclecloud.com/{solution-customer-env}/retailhome
   ```

2. Navigate to **Settings** → **Application Administration** → **Customer Modules Management**. Confirm the table on this page loads without error and displays multiple rows of results. If an error occurs, contact Oracle Support.



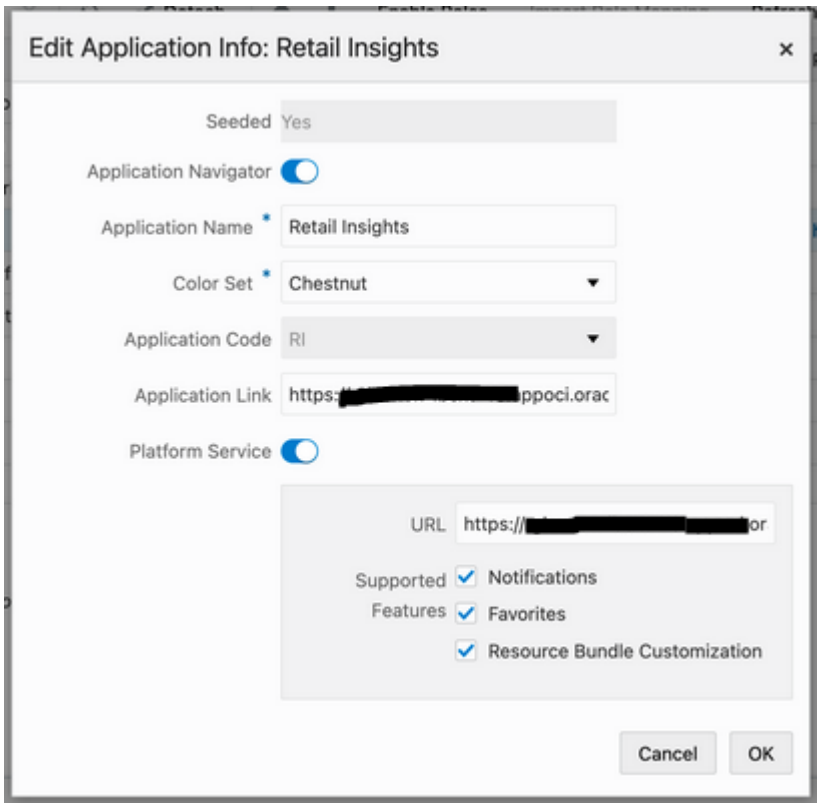
3. You may enable or disable various modules, depending on your implementation plans. For example, if you are not implementing any Retail Insights modules, then the sections for “RCI” and “RMI” can be deactivated.

> **Note:**
>
> Other components within the RI parent module may still be necessary. Detailed module requirements are described in Batch Orchestration

In addition to Customer Modules, you may also use Retail Home's Resource Bundle Customization (RBC) feature to change translatable strings in the applications to custom values. Use the steps below to verify this feature is available:

1. Navigate to **Settings** → **Application Administration** → **Application Navigator Setup**.
2. Confirm that a row already exists for each application in the platform, including Retail Insights, Retail Science Platform, and Merchandise Financial Planning.
3. On Retail Insights, select the row and click **Edit**.
   a. If not enabled, change the **Platform Service** toggle to an enabled state.
   b. Check all of the boxes that appear.
   c. Enter a valid platform service **URL**.

      If your platform services URL is blank and you do not know the URL, log a Service Request to receive it from Oracle.
4. Repeat the steps above for the RSP and MFP modules, if necessary.



5. Navigate to **Settings** → **Resource Bundles** → **Resource Text Strings** once the navigator and platform service setup is validated.

6. Set the following values in the dropdown menus:
   a. **Application:** Retail Insights
   b. **Bundle:** Retail Insights
   c. **Language:** AMERICAN (en)

7. Click the **Search** button and wait for results to return.
   - If multiple rows of results are returned, then you have successfully verified the feature is enabled and functioning properly.
   - If you receive an error, contact Oracle Support.

If you need additional details on how the RBC feature is used within each application module, refer to the product-specific documentation sets, such as the *Retail Insights Administration Guide*.

Process Orchestration and Monitoring (POM)

The Process Orchestration and Monitoring (POM) application is a user interface for scheduling, tracking, and managing both nightly as well as intraday batch jobs for applications such as RI, MFP, and RSP. Two important screens from the POM application are Batch Monitoring and Batch Administration. The Batch Monitoring window provides a runtime view of the statuses and dependencies of the different batch cycles running on the current business day. Batch Administration allows you to modify the batch schedules and synchronize them with Retail Home.

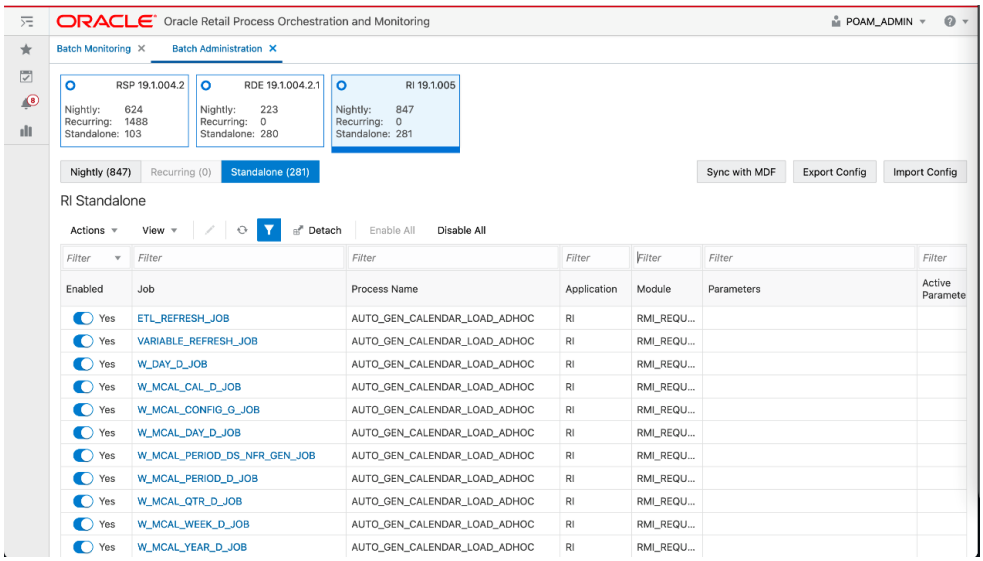
For general information about POM and the features it provides, review the *POM Implementation Guide* and *POM User Guides*.

POM and Customer Modules Management

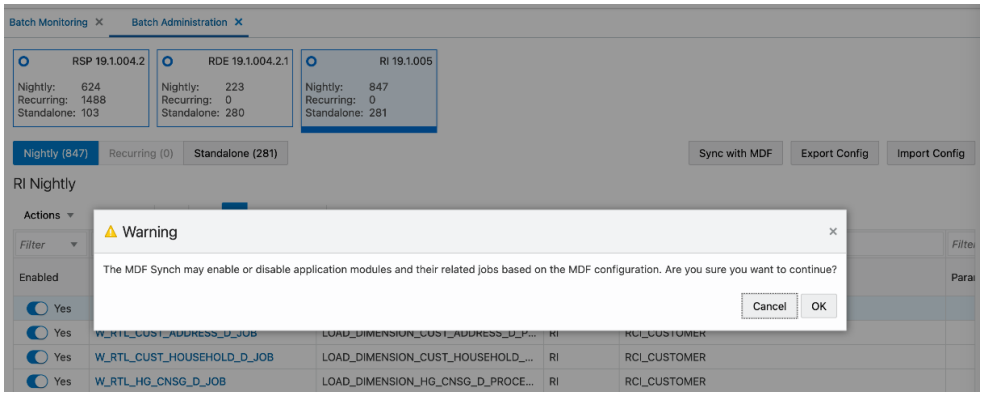
A required implementation step for RAP will be to synchronize customer modules from Retail Home to POM to set up your starting batch processes and turn off any processes you are not using. The steps below explain the general process for syncing POM and Retail Home, which are used by Retail Insights and Science applications to set up the nightly batches. They are also required for the RAP common components used by all the modules.

1. Log in to the Retail Home application as a user with the `RETAIL_HOME_ADMIN` (or `RETAIL_HOME_ADMIN_PREPROD`) user role.
2. Navigate to **Settings** → **Application Administration** → **Customer Modules Management**.
3. Configure your batch modules as needed, disabling any components which you do not plan to implement, then click the **Save** button to complete the setup.

4. Now log in to the POM application URL with a user granted the `BATCH_ADMINISTRATOR_JOB` or pre-prod equivalent role.
5. Navigate to **Tasks → Batch Administration**.
6. Click on the tile named **RI <Release_#>** to view the RI batch jobs, which should be loaded into the table below the tiles.



7. Click the **Sync with MDF** button (above the table) and then click the **OK** button in the Warning message popup. Once clicked, the Platform Services calls are initiated between Retail Home and POM to sync the module status.

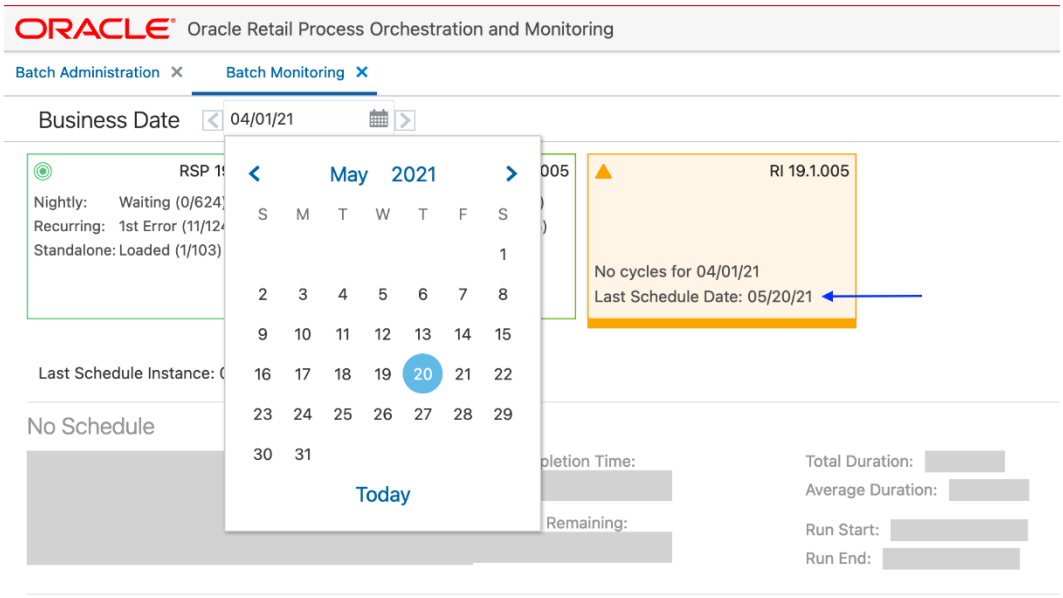


8. While the modules are synchronizing, you will see a message: 'Some features are disabled while a schedule is being synced'. Do not attempt to modify the schedule or make other changes while the sync is in progress.

9. Once the sync is complete, a JSON file with the batch schedule summary is downloaded. This file contains the current and previous status of an application and module in MDF and POM after sync. For example:

{"scheduleName":"RI","synced":true,"enabledModules":[{"state":"MATCHED_MODULE","mdfStatus":"ENABLED","prevMdfStatusInPom":"ENABLED","prevStatusInPom":"ENABLED","publishToPom":true,"applicationName":"RI","moduleName":"RMI_SI_ONORDER","matchedModule":true},...

10. Click the **Nightly** or the **Standalone** tab above the table and enter a filter for the Module column (based on the modules that were activated or deactivated) and press `Enter`. The jobs will be enabled or disabled based on the setup in Customer Modules Management.
11. Navigate to **Tasks** → **Batch Monitoring**. Click on the same application tile as before. If the batch jobs are not listed, change the **Business Date** option to the 'Last Schedule Date' shown on the tile.



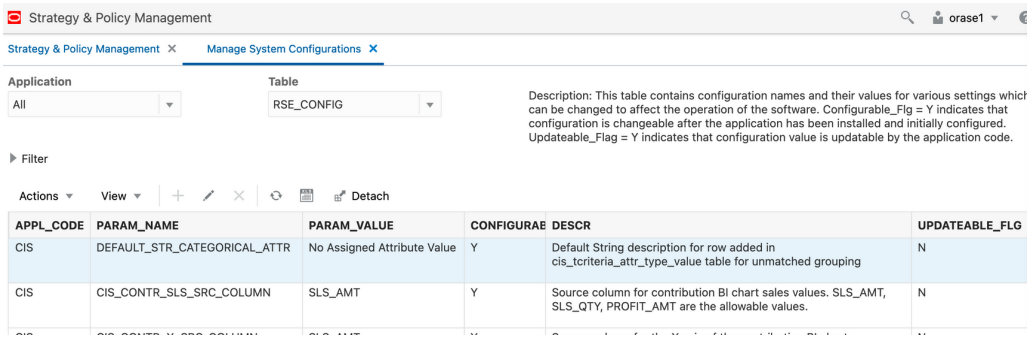
12. Once the date is changed, the batch jobs are loaded in the table. Click the **Restart Schedule** button so that module changes are reflected in the new schedule. Click **OK** on the confirmation pop-up. After a few seconds, a 'Restarted' message is displayed.
13. In the same screen, filter the Job column (for example,'W_HOUSEHOLD') to check the status of jobs. The status is either 'Loaded' or 'Disabled' based on the configuration in the Customer Modules Management screen in Retail Home.

> **Note:**
>
> A specific module in Retail Home may appear under several applications, and jobs within a module may be used by multiple processes in POM. The rule for synchronizing modules is, if a given POM job is enabled in at least one module, it will be enabled in POM (even if it is disabled in some other modules). Only when a job is not needed for any modules will it be disabled.

Control & Tactical Center

Retail Insights and Science modules make use of a centralized configuration interface named the Control & Tactical Center. From here the user can review and override the system configurations for different applications through the Manage System Configurations screen.

The table can be filtered by Application and their configured tables. There is also a Description section on the right side that displays the details of the filtered table.



Here are the steps for accessing and using this feature:

1. To access the system configurations, start from the Retail Home URL sent to your cloud administrator on first provisioning a new environment. It should look similar to the URL format below.

   ```
   https://{service}.retail.{region}.ocs.oraclecloud.com/{solution-customer-env}/retailhome
   ```

2. Using the Retail Home application menu, locate the link for the Retail Science Platform (RSP). Alternatively, you can directly navigate to the application using a URL similar to the format below.

   ```
   https://{service}.retail.{region}ocs.oraclecloud.com/{solution-customer-env}/orase/faces/Home
   ```

3. In the task menu, navigate to **Control & Tactical Center → Strategy & Policy Management**. A new window opens.

   > **Note:**
   >
   > Make sure your user has the `ADMINISTRATOR_JOB` role in IDCS or OCI IAM before logging into the system.

4. Click **Manage System Configurations** in the new application screen.
5. Select an application in the dropdown menu to pick the desired set of configurations. Based on the selection, the Filter and Table options are populated with the configured columns and data. The Description section also displays the details of the selected table.

Specifically for the initial environment setup, you will be working mainly within the Retail Insights group of configuration tables. You will also use the Strategy & Policy Management interface to access the forecasting configurations needed to set up and generate forecasts for Planning applications. It is also used to manage the business policies and rules used by Offer Optimization. Any required configurations in these areas will be covered later in this document.

Data Visualizer

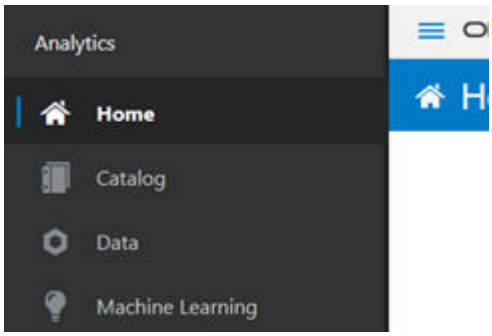
Retail Analytics Platform implementations largely involve processing large volumes of data through several application modules, so it is important to know how to access the database to review settings, monitor load progress, and validate data tables. Database access is provided through the Oracle Data Visualization (DV) tool, which is included with all Retail Analytics Platform environments. The URL to access the DV application will be similar to the below URL:

```
https://{service}.retail.{region}ocs.oraclecloud.com/{solution-customer-env}/dv/?pageid=home
```

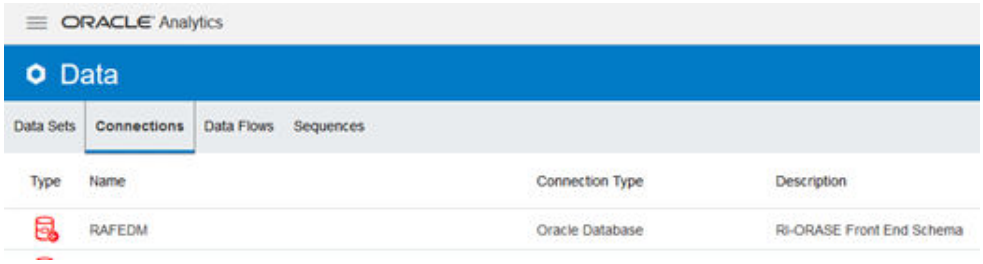
> **Note:**
>
> The best way to write ad hoc SQL against the database is through APEX. However, Data Visualizer can be used to create reusable datasets based on SQL that can be built into reports for longer term usage.

The RAP database comprises several areas for the individual application modules, but the majority of objects are exposed in DV as a connection to the `RAFEDM01` database user. This user has read-only access to the majority of database objects which are involved in RI and Science implementations, as well as the tables involved in publishing data to the Planning modules. Follow the steps below to verify access to this database connection:

1. Log in to the DV application with a user that includes the `RIApplicationAdministrator_JOB` or `DVContentAuthor` groups in IDCS or OCI IAM (in non-production environments, use `RIApplicationAdministrator_JOB_PREPROD` or `DVContentAuthor`).
2. Expand the navigation panel using the Navigator icon in the upper left corner.

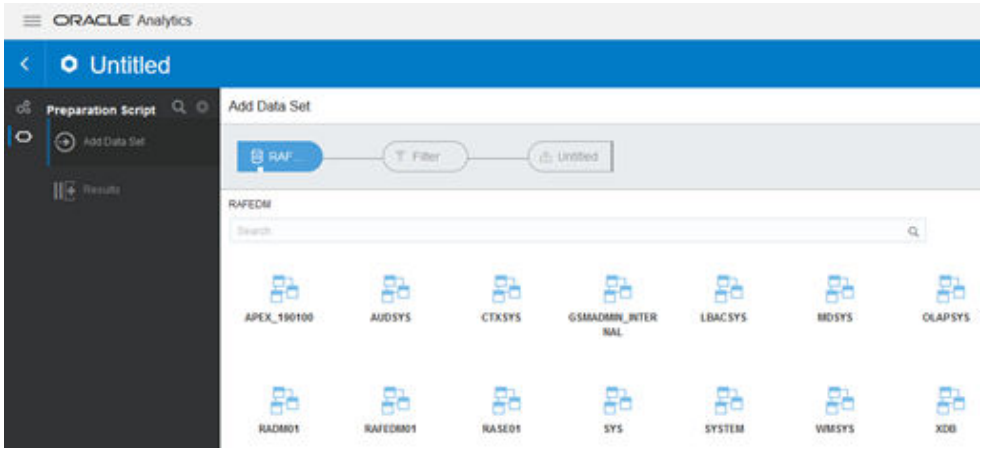


3. Click **Data** and, once the screen loads, click **Connections**. Confirm that you have a connection already available for RAFEDM (Retail Analytics Front End Data Mart).

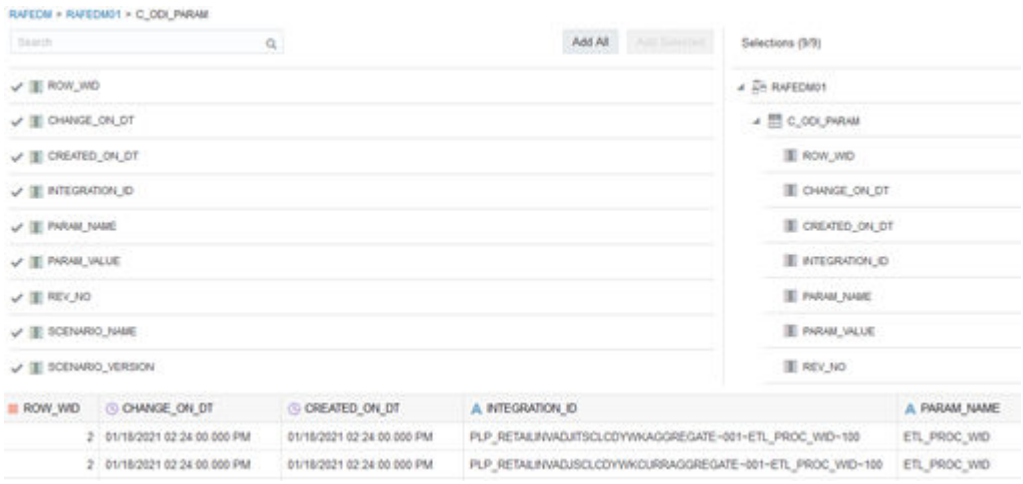


4. Click the connection. The Add Data Set screen will load using the selected connection. A list of database users are displayed in the center panel.

   If any errors are displayed or a password is requested, contact Oracle Support for assistance.



5. Click the **RAFEDM01** user.
6. Click **C_ODI_PARAM** on the following list of database tables.
7. Click the **Add All** button to select all columns in the table. In the bottom panel, click the **Refresh** icon to receive sample data. Confirm that one or more rows of data are displayed.

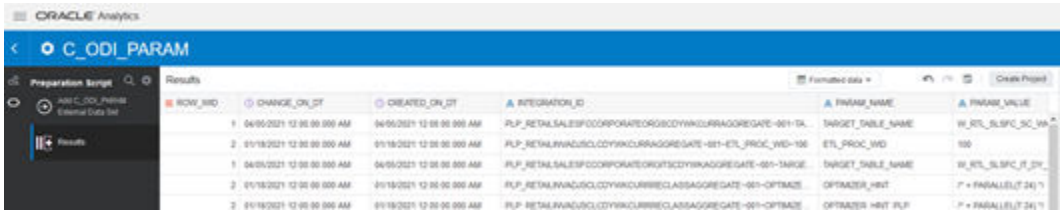


8. If you are performing a one-time query that does not need to be repeated or reused, you can stop at this point. You can also switch to the **Enter SQL** option in

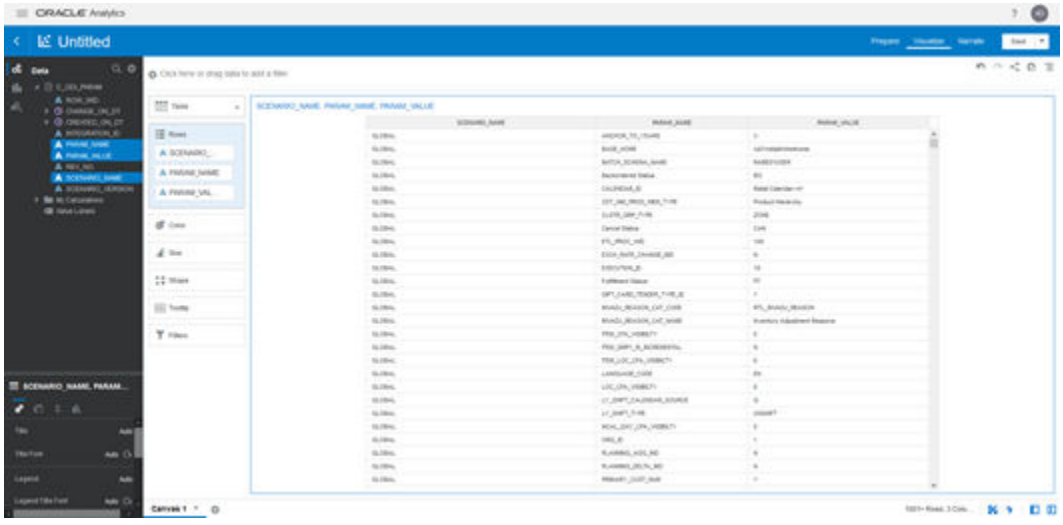
the upper right corner, to write simple queries on the database. However, if you want to create a reusable dataset, or expose the data for multiple users, proceed to the next steps.

> **Note:**
>
> The Enter SQL option does NOT allow for the full range of Oracle SQL commands. You cannot join multiple tables or perform complex operations such as pivots and partition-by clauses. The primary purpose is to select columns to add to a dataset and do basic manipulations of them.

9. Click the last icon in the data flow at the top of the screen.
10. Change the value of the **Data Access** setting to `Live`.
11. Click **Add** to complete the dataset definition and start the dataset formatting process.



12. You can format the dataset on this screen for use in DV projects. You may rename the columns, change the datatype between Measure and Attribute, create new columns based on calculated values, and extract values from existing columns (such as getting the month from a date). Refer to Oracle Analytics documentation on Dataset creation for full details. When finished, click **Create Project** in the upper right corner to save the dataset and open a new project with it.



Once you have verified database connectivity, you may continue on to creating more datasets and projects as needed. Datasets will be saved for your user and can be reused at later dates without having to re-query the database. Saved datasets can be accessed using the Data screen from the Navigator panel.

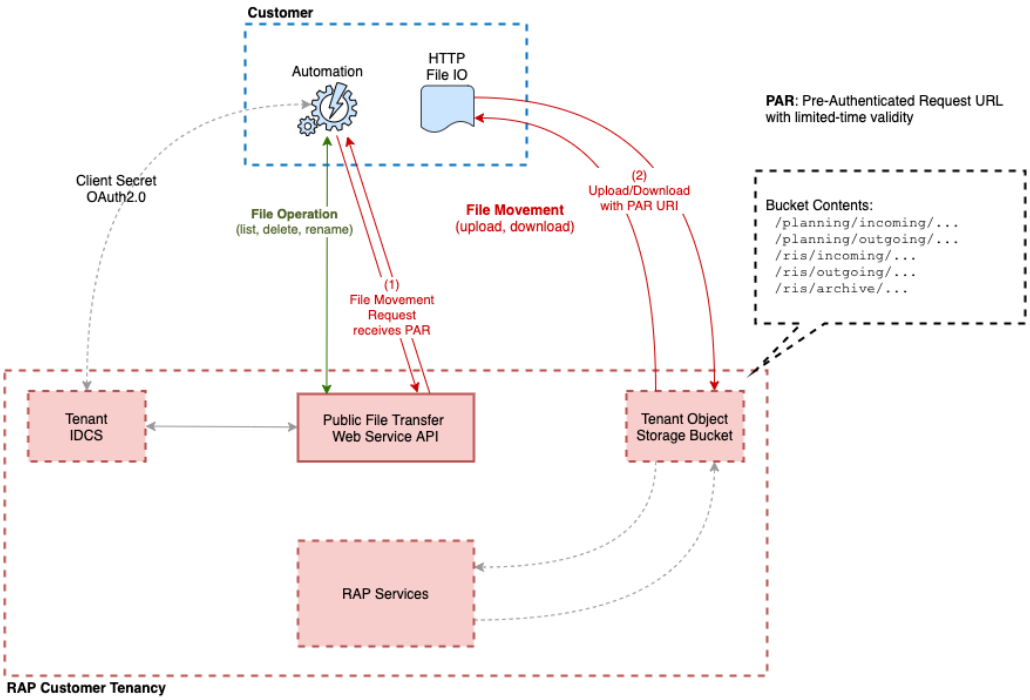
Object Storage

File Transfer Services (FTS) for the Retail Analytics Platform cloud services are being made available in this release, replacing SFTP in new environments. They will allow you to manage uploading and downloading files to Oracle Cloud Infrastructure Object Storage, which is an internet-scale, high-performance storage platform that offers reliable and cost-efficient data durability.

Access to files is through a pre-authenticated request (PAR), which is a URL that requires no further authentication to upload or download files to the cloud. To retrieve a PAR, you must use the appropriate file transfer REST service. These new services will enable you to import files to and export files from Object Storage used by the solutions. The primary role of these services is to ensure that only valid external users can call the service by enforcing authorization policies. Where supported, the files can be compressed (zipped) before upload.

The general flow of activities involving FTS and an external integrating system are as follows for transferring files into our cloud service:

1. The third-party solution calls the service, requesting pre-authentication to upload files from Object Storage, including the incoming prefix and file name. On receiving the PAR, the file is uploaded using the URL included within the response. A PAR has a validity of 10 minutes; an upload can take longer than 10 minutes but after it is returned it must be started within that period.
2. The cloud service batch processing will retrieve the files from Object Storage, after they have passed an anti-virus and malware scan.
3. The batch processing will delete the file from Object Storage to ensure it is not re-processed in the next batch run. The batch processing uncompresses the file and processes the data.



To interact with FTS you must use the REST APIs provided. The table below lists the API end points for different file operations.

Operation	Method	FTS API Endpoint
Ping	GET	`{baseUrl}/services/private/FTSWrapper/ping`
List Prefixes	GET	`{baseUrl}/services/private/FTSWrapper/listprefixes`
List Files	GET	`{baseUrl}/services/private/FTSWrapper/listfiles`
Move Files	POST	`{baseUrl}/services/private/FTSWrapper/movefiles`
Delete Files	DELETE	`{baseUrl}/services/private/FTSWrapper/delete`
Request Upload PAR	POST	`{baseUrl}/services/private/FTSWrapper/upload`
Request Download PAR	POST	`{baseUrl}/services/private/FTSWrapper/download`

The `{baseUrl}` is the URL for your RAP service supplied to you when your service is provisioned. Refer to the Required Parameters section for additional parameters you will need to make FTS requests.

Required Parameters

To leverage File Transfer Services, several pieces of information are required. This information is used in API calls and also inserted into automated scripts, such as the test script provided later in this document.

The below parameters are required for uploading data files to object storage.

```
BASE_URL="https://__YOUR_TENANT_BASE_URL__"
TENANT="__YOUR-TENANT_ID__"
IDCS_URL="https://_YOUR__IDCS__URL__/oauth2/v1/token"
IDCS_CLIENTID="__YOUR_CLIENT_APPID__"
IDCS_CLIENTSECRET="__YOUR_CLIENT_SECRET___"
IDCS_SCOPE="rgbu:rsp:psraf-__YOUR_SCOPE__"
```

Base URL

The substring before the first '/' in the Application URL is termed as the base URL.

Example URL: `https://rap.retail.eu-frankfurt-1.ocs.oraclecloud.com/rgbu-rap-hmcd-stg1-rsp/orase/faces/Home`

In the above URL, `BASE_URL` = `https://rap.retail.eu-frankfurt-1.ocs.oraclecloud.com`

Tenant

The string after the Base URL and before the Application URL starts would be the Tenant.

Example URL: `https://rap.retail.eu-frankfurt-1.ocs.oraclecloud.com/rgbu-rap-hmcd-stg1-rsp/orase/faces/Home`

In the above URL, `TENANT` = `rgbu-rap-hmcd-stg1-rsp`

IDCS or OCI IAM URL

Your authentication URL is the one used when you first access any of your cloud services and are redirected to a login screen. You will get the base URL from the login screen and combine it with the necessary path to fetch the authentication token.

Example Base URL: `https://idcs-a4cbf187f29d4f41bc03fffb657d5513.identity.oraclecloud.com/`

Using the above URL, `IDCS_URL` = `https://idcs-a4cbf187f29d4f41bc03fffb657d5513.identity.oraclecloud.com/oauth2/v1/token`

IDCS or OCI IAM Scope

The authentication scope is a code associated with the specific environment you are planning to interact with. It has a static prefix based on the application, appended with an environment-specific code and index.

Base format: `rgbu:rsp:psraf-<ENV><ENVINDEX>`

Where `<ENV>` is replaced with one of the codes in (`PRD`, `STG`) and `<ENVINDEX>` is set to 1, unless you have multiple staging environments, and then the index can be 2 or greater.

To determine this information, look at the URL for your environment, such as:

`https://rap.retail.eu-frankfurt-1.ocs.oraclecloud.com/rgbu-rap-hmcd-`**`stg1`**`-rsp/orase/faces/Home`

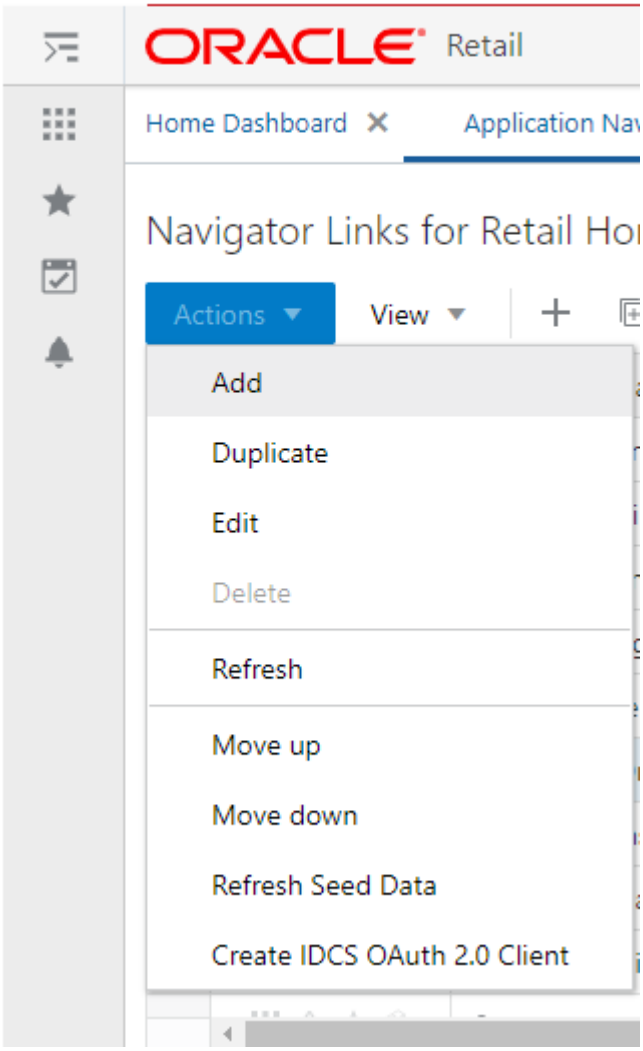
In the tenant string, you can see the code `stg1`. This can be added to your scope string (ensuring it is in uppercase characters only).

`IDCS_SCOPE` = `rgbu:rsp:psraf-STG1`

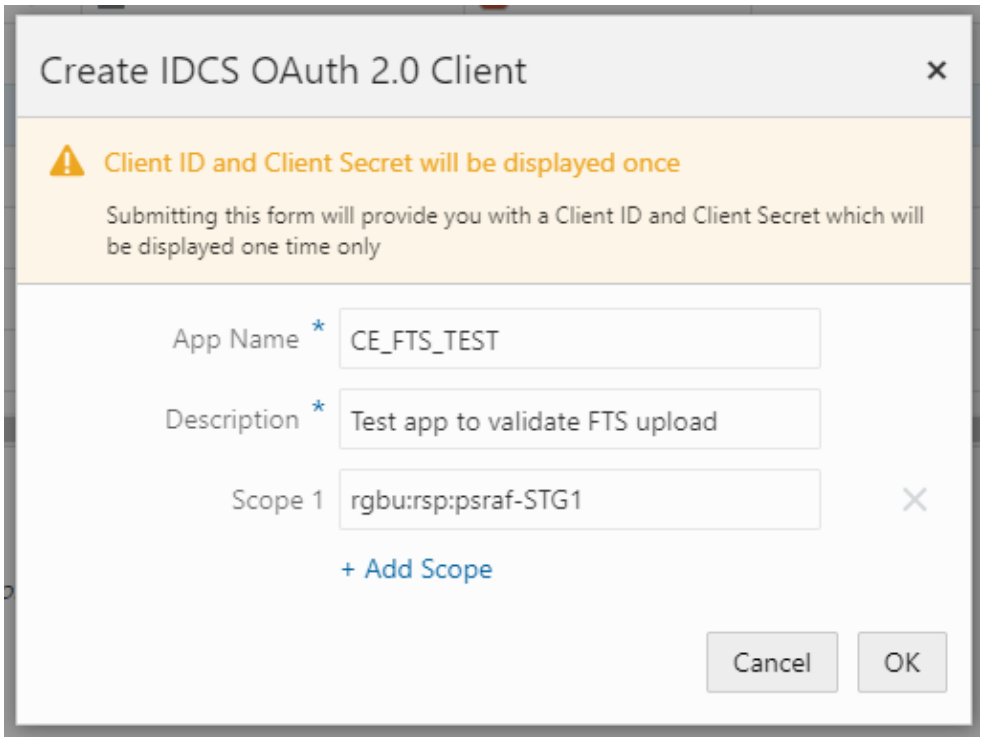
Client ID and Secret

The client ID and secret are authentication keys generated for your specific connection and must be passed with every request. Retail Home provides an interface to get these values when you login with a user having the `PLATFORM_SERVICES_ADMINISTRATOR` group.

1. Navigate to the Application Navigator Setup screen.
2. Select a row for the application you will transfer files to. The row must already have a Platform Services URL assigned to it. Use the Actions menu to select the **Create IDCS OAuth 2.0 Client** action.



3. Enter the requested details in the window. The application name should be unique to the connection you are establishing and cannot be used to generate multiple client ID/secret pairs.



The application name cannot be re-used for multiple requests. It also cannot contain spaces. The scope should be the string previously established in IDCS or OCI IAM Scope. The description is any value you wish to enter to describe the application name being used.

4. Click **OK** to submit the form and display a new popup with the client ID and secret for the specified Application Name. Do NOT close the window until you have captured the information and verified it matches what is shown on screen. Once you close the window, you cannot recover the information and you will need to create a new application.

Common HTTP Headers

Each call to FTS should contain the following HTTP headers:

```
Content-Type: application/json
Accept: application/json
Accept:-Language: en
Authorization: Bearer {ClientToken}
```

The `{ClientToken}` is the access token returned by IDCS or OCI IAM after requesting client credentials. This is refreshed periodically to avoid authentication errors.

Retrieving Identity Access Client Token

The access client token is returned from a POST call to the IDCS or OCI IAM URL, provided at provisioning, along with the following:

Headers
`Content-Type: application/x-www-form-urlencoded`
`Accept: application/json`
`Authorization: Basic {ociAuth}`

`ociAuth` is the base64 encoding of your `{clientId}:{clientSecret}`

Data (URLEncoded)
`grant_type=client_credentials`
`scope=rgbu :rpas :psraf-{environment}`

FTS API Specification

An example shell script implementing these API calls can be found in Sample Public File Transfer Script for Planning Apps. The sample script will require all of the parameters discussed so far in this chapter to be added to it before it can be used to issue FTS commands. Refer to the table below for a more detailed list of services available in FTS.

Ping	Returns the status of the service, and provides an external health-check.
Method	GET
Endpoint	`{baseUrl}/services/private/FTSWrapper/ping`
Parameters	Common headers
Request	None
Response	`{ appStatus:200 }` The appStatus code follows HTTP return code standards.

List Prefixes	Returns a list of the known storage prefixes. These are analogous to directories, and are restricted to predefined choices per service.
Method	GET
Endpoint	`{baseUrl}//services/private/FTSWrapper/listprefixes`
Parameters	Common headers
Request	None
Response	A JSON array of strings containing the known prefixes.

List Files	Returns a list of the file within a given storage prefix.
Method	GET
Endpoint	`{baseUrl}/services/private/FTSWrapper/listfiles`
Parameters	Common headers

Request	Query parameters (`…/listfiles?{parameterName}`) that can be appended to the URL to filter the request: `prefix` – the storage prefix to use `contains` – files that contain the specified substring `scanStatus` – file status returned by malware/antivirus scan `limit` – control the number of results in a page `offset` – page number `sort` – the sort order key
Response	A JSON `resultSet` containing array of files. For each file, there is metadata including: name, size, created and modified dates, scan status and date, scan output message.

Move Files	Moves one or more files between storage prefixes, while additionally allowing the name to be modified
Method	GET
Endpoint	`{baseUrl}/services/private/FTSWrapper/movefiles`
Parameters	Common headers

Request	An array of files containing the current and new storage prefixes and file names, as shown below.
	
	`{"listOfFiles": [`
	`{"currentPath":`
	`{ "storagePrefix": "string",`
	`"fileName": "string"},`
	`"newPath": {`
	`"storagePrefix": "string",`
	`"fileName": "string"`
	`}`
	`}`
	`]`
	`}`

Response	HTTP 200, request succeeded; HTTP 500, an error was encountered.

Delete Files	Deletes one more or files
Method	DELETE
Endpoint	`{baseUrl}/services/private/FTSWrapper/deletefiles`
Parameters	Common headers

Request	A JSON array of files to be deleted. One or more pairs of `storagePrefix` and `filename` elements can be specified within the array.
	
	`{"listOfFiles":`
	`[`
	`{`
	`"storagePrefix": "string",`
	`"fileName": "string"`
	`}`
	`]`
	`}`

Response	A JSON array of each file deletion attempted and the result.

Request Upload PAR	Request PAR for uploading one or more files
Method	POST
Endpoint	`{baseUrl}/services/private/FTSWrapper/upload`
Parameters	Common headers

Request	A JSON array of files to be uploaded. One or more pairs of `storagePrefix` and `filename` elements can be specified within the array.
	
	`{ "listOfFiles":`
	`[`
	`{`
	`"storagePrefix": "string",`
	`"fileName": "string"`
	`}`
	`]`
	`}`

Response	A `parList` containing an array containing elements corresponding to the request including the PAR `accessUri` and name of file.

Request Download PAR	Request PARs for downloading one or more files
Method	POST
Endpoint	`{baseUrl}/services/private/FTSWrapper/download`
Parameters	Common headers

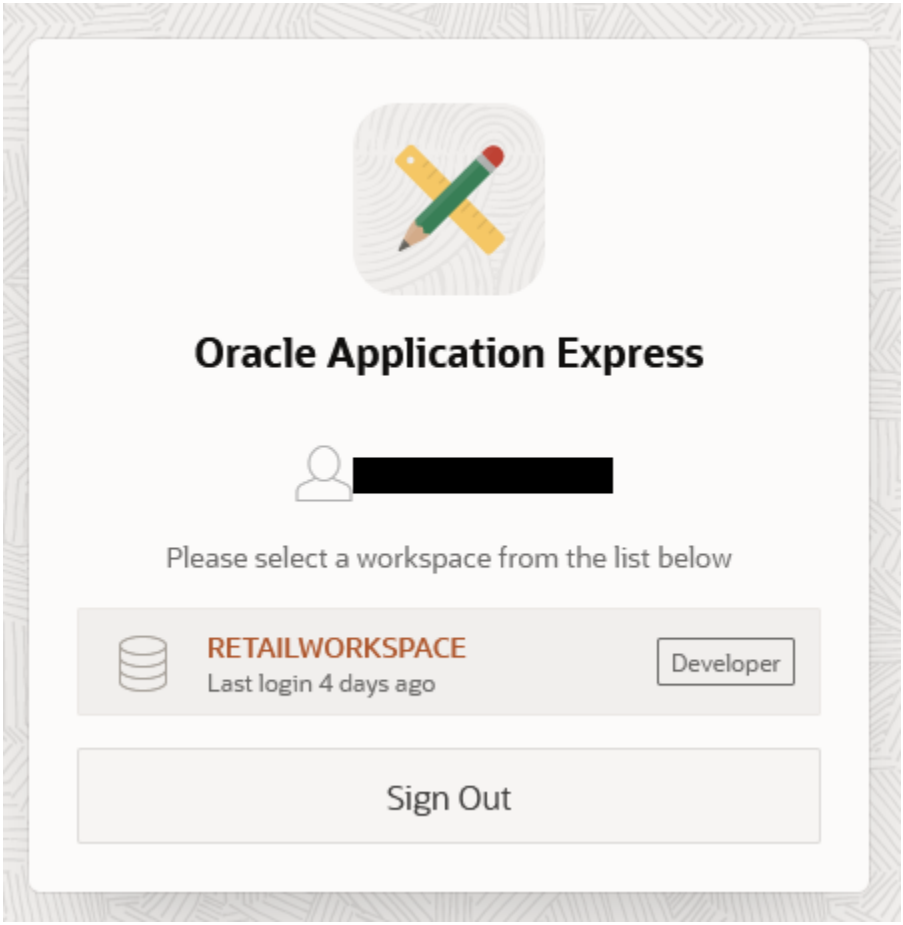
Request	A JSON array of files to be downloaded. One or more pairs of storagePrefix and filenames can be specified within the array. `{ "listOfFiles": [ { "storagePrefix": "string", "fileName": "string" } ] }`
Response	A `parList` containing an array containing elements corresponding to the request including the PAR `accessUri` and name of file.

Application Express (APEX)

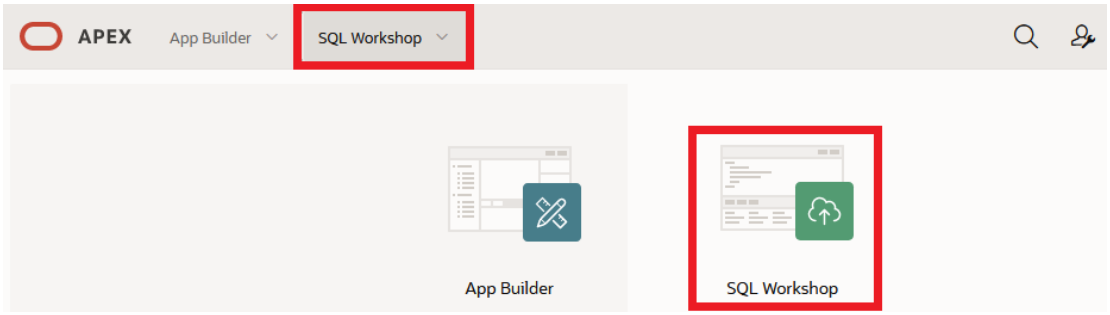
The Retail Analytics Platform includes a dedicated instance of Application Express (APEX) which will be pre-configured for read-only database access to RAP tables and views. APEX is managed as a part of the Innovation Workbench (IW) module which includes APEX and Data Studio. You can access APEX from a task menu link in the Science Platform interface, or by navigating directly to the ORDS endpoint like below:

```
https://{service}.retail.{region}.ocs.oraclecloud.com/{solution-customer-env}/ords
```

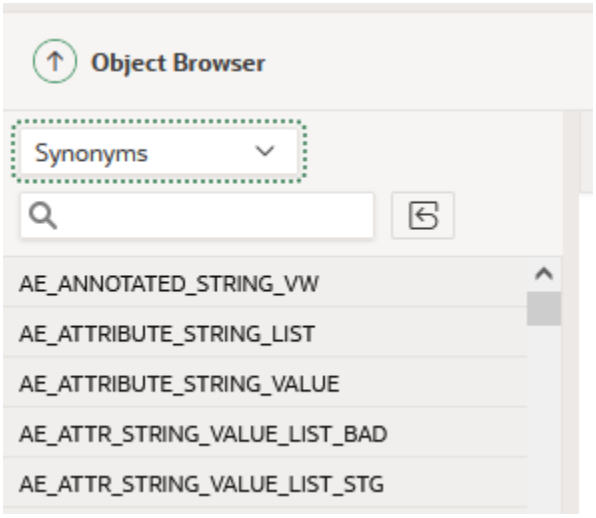
Once you access the URL and after entering your Single Sign On information, you should be presented with an APEX login screen. From here you will select your workspace, which is how the system knows which database schema to access.



If you are not presented with a workspace to select, contact Oracle Support for assistance. Click the name of the workspace (`RETAILWORKSPACE` in the above example) to sign into APEX. Once logged in, click on the **SQL Workshop** icon or access the **SQL Workshop** menu to enter the SQL Commands screen. This screen is where you will enter SQL to query the RAP database objects in the RI and Science schemas.



To see the list of available objects to query, access the **Object Browser**. All RI and Science objects are added as synonyms, so select that menu option from the panel on the left.



To see the available columns on these tables, select all columns from the **SQL Commands** screen and review the results. If the table does not yet have data, then you can also locate the table definition from Data Visualizer's Dataset creation flow by accessing the `RAFEDM01` user and locating the table definition within the resulting list of objects. If neither option is sufficient to get the information you need, contact Oracle Support for further assistance.

Postman

For automated API calls, Postman is the preferred way to interact with POM in order to call ad hoc processes and perform data load activities. The steps below explain how to configure Postman for first-time use with POM.

> **Note:**
>
> POM versions earlier than v21 allow Basic Authentication, while v21+ requires OAuth 2.0 as detailed below.

1. As a pre-requisite, retrieve the Client ID and Client Secret from Retail Home's 'Create IDCS OAuth 2.0 Client'. Refer to the *Retail Home Administration Guide* for complete details on retrieving the Client ID and Client Secret info if you have not done it before.
2. In the Postman application, click **New->HTTP Request**.
3. Set Request Type as `POST` and set the Request URL (Example for RI schedule):

   ```
   https://<Region-LB>/<POM-Subnamespace>/ProcessServices/services/
   private/executionEngine/schedules/RI/execution
   ```

For example:

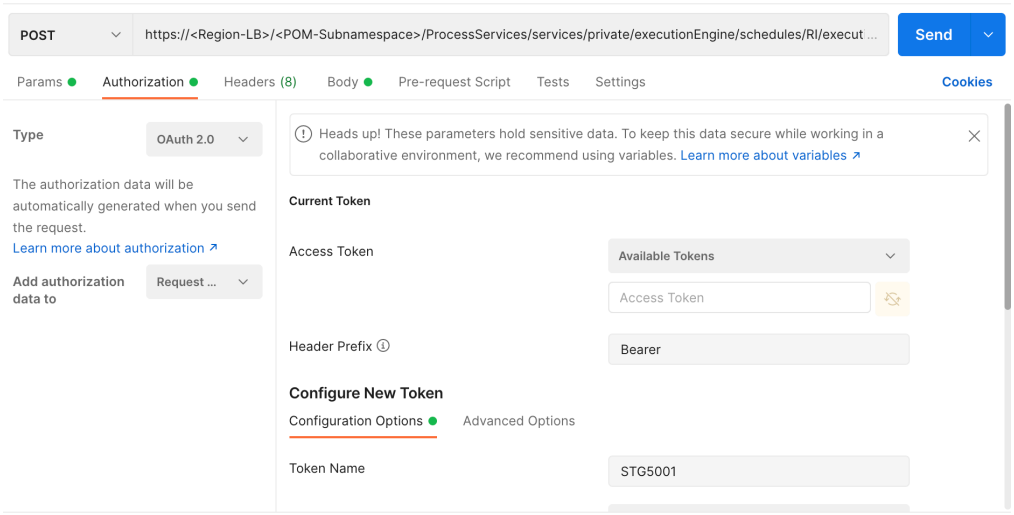
```
https://home.retail.us-region-1.ocs.oc-test.com/rgbu-common-rap-prod-pom/
ProcessServices/services/private/executionEngine/schedules/RI/execution
```

4. Perform the following steps before sending the POST request to retrieve your authentication token:

   a. Authorization Tab:

      Type: `OAuth 2.0`

      Add authorization data to: `Request Headers`



   b. Configure the New Token:

      i. **Token Name:** `PROD5555`

      ii. **Grant Type:** `Client Credentials`

      iii. **Access Token URL:** `https://<Customer-IDCS>/oauth2/v1/token`

      For example: `https://idcs-fe5f77f8a44.identity.c9dev.oc9dev.com/oauth2/v1/token`

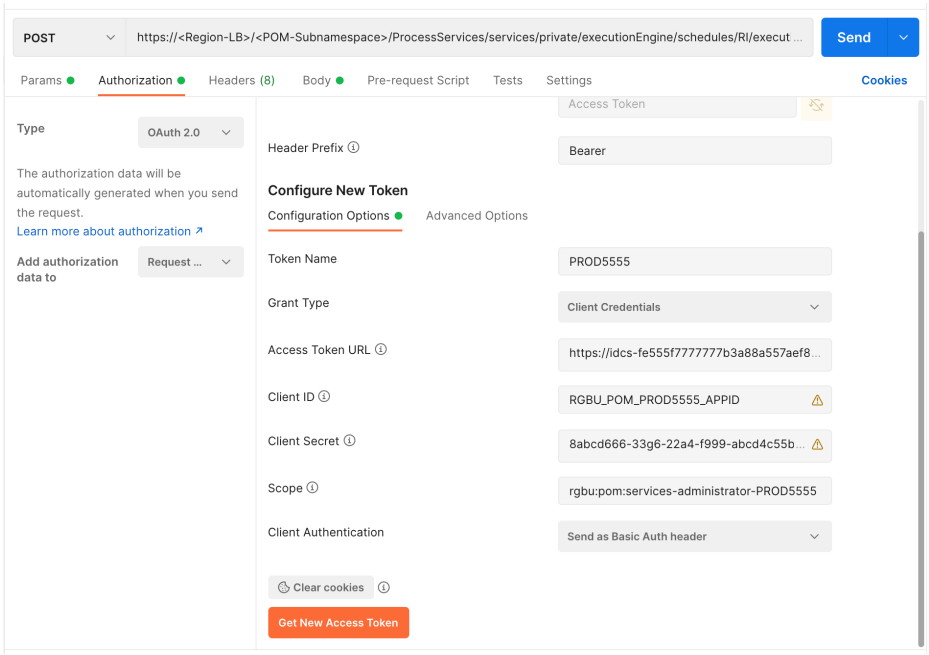
      iv. **Client ID:** A unique “API Key” generated when registering your application in the Identity Cloud Services admin console.

      v. **Client Secret:** A private key similar to a password that is generated when registering your application in the Identity Cloud Services admin console.

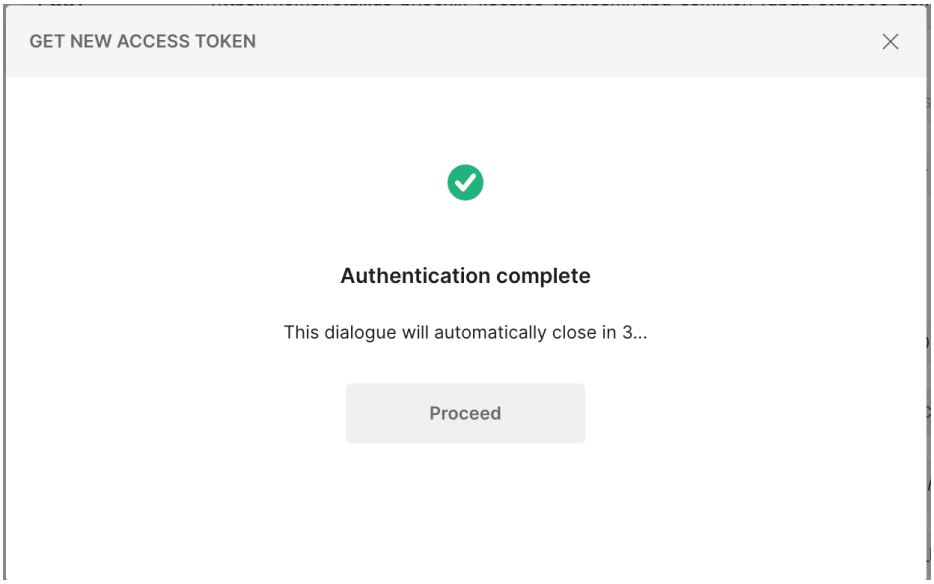
      vi. **Scope:** `rgbu:pom:services-administrator-<Env-INDEX>`

      For example: `rgbu:pom:services-administrator-PROD5555`

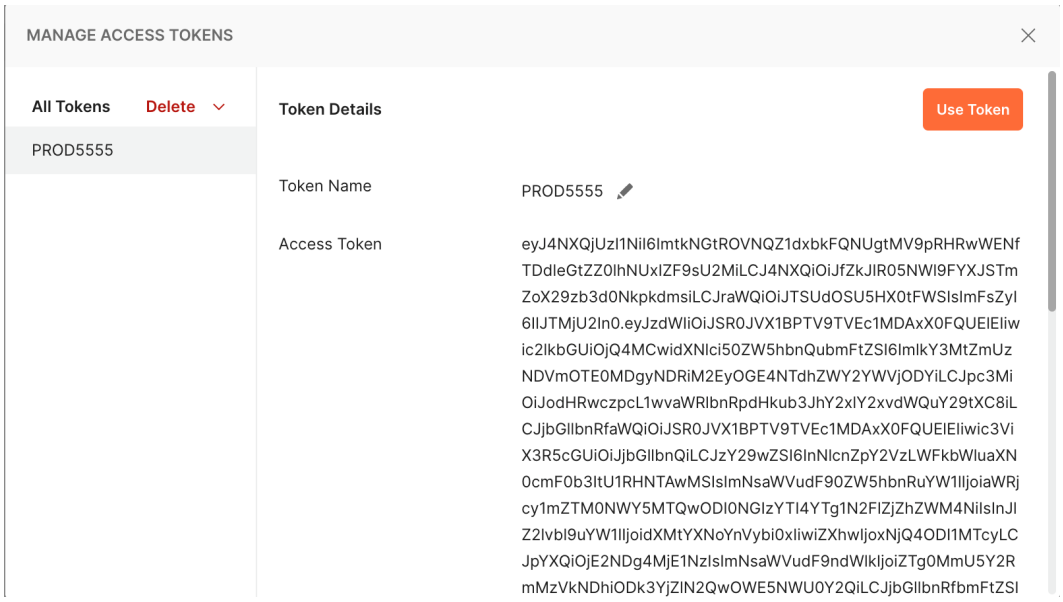
      vii. **Client Authentication:** `Send as Basic Auth header`



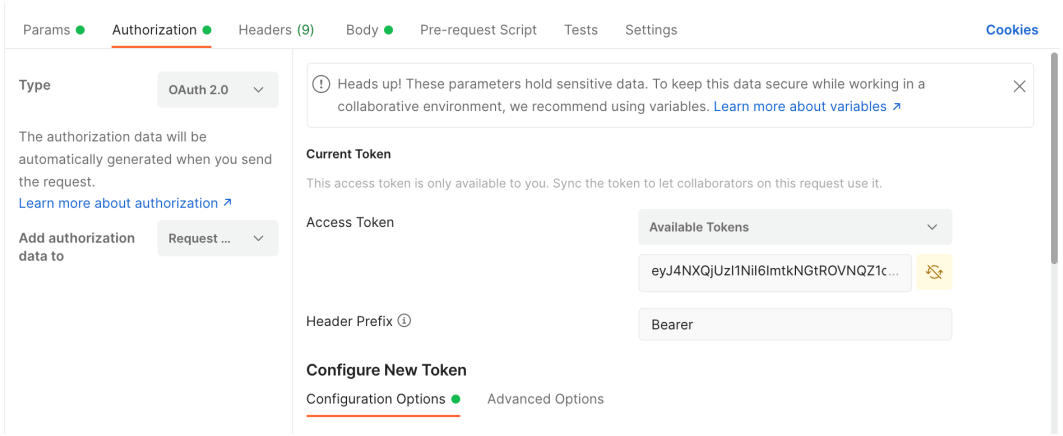
c. Click the **Get New Access Token** button.



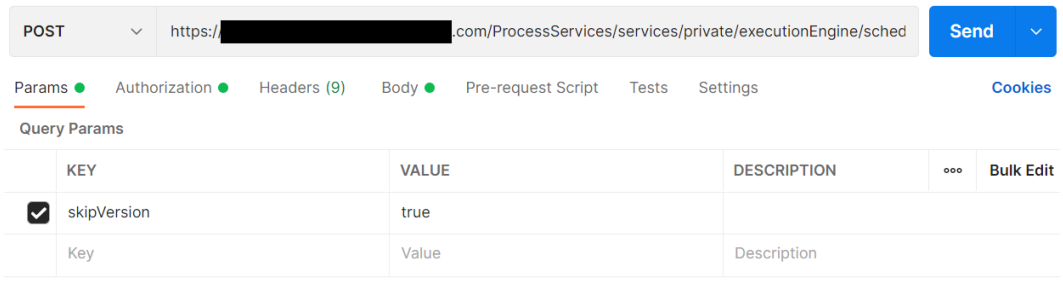
5. The Access Token is displayed in the **MANAGE ACCESS TOKENS** pop-up window. Click the **Use Token** button.



6. The generated token is populated in the **Access Token** section.



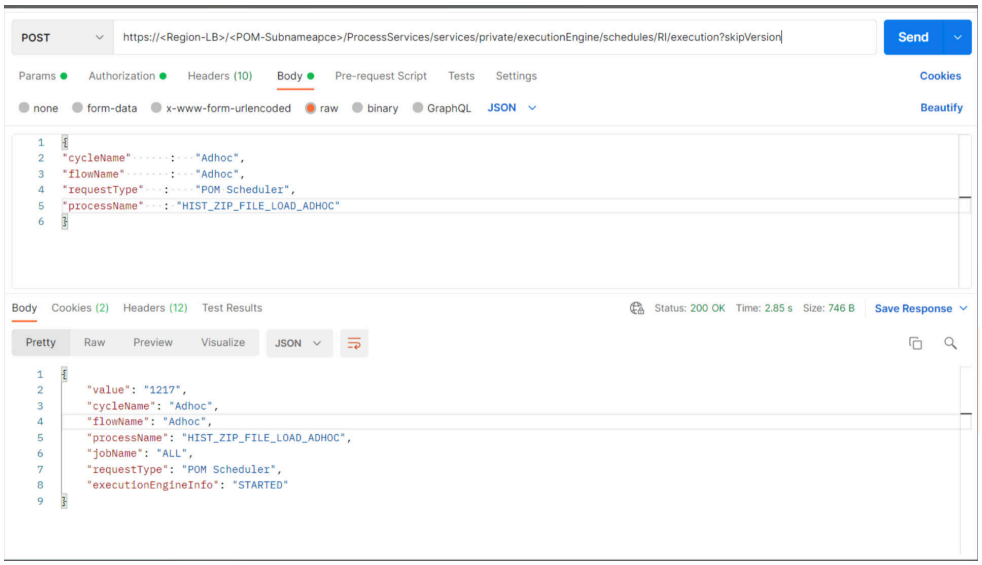
7. In the **Parameters**, provide a key of `skipVersion` with a value of `true`.



8. Click the **Body** tab to enter the JSON for running POM batches.
9. Select the **raw** radio button and **JSON** file type.
10. Enter the JSON XML and click the **Send** button in the **Body** tab.
    a. If the returned Status is "200 OK" and `'executionEngineInfo': "STARTED"` is in the response body, then the batch started as expected.

b. If the status is not 200 and either of 401 (authorization error) or 500 (incorrect body content) or 404 (server down) and so on, then perform error resolution as needed.

Example request and response for the `HIST_ZIP_FILE_LOAD_ADHOC` process:



Additional examples of Postman Body JSON XML are listed below.

Nightly Batch

```
{
"cycleName"    :  "Nightly",
"flowName"     :  "Nightly",
"requestType"  :  "POM Scheduler"
}
```

With One Process – Applicable for Nightly and Ad Hoc

```
{
"cycleName"     :  "Nightly",
"flowName"      :  "Nightly",
"requestType"   :  "POM Scheduler",
"processName"   :  "LOAD_AGGREGATION_BCOST_IT_DY_A_PROCESS"
}
```

With Dynamic Parameters – Applicable for Nightly and Ad Hoc

```
{
"cycleName"         :  "Adhoc",
"flowName"          :  "Adhoc",
"requestType"       :  "POM Scheduler",
"processName"       :  "HIST_INVRTV_LOAD_ADHOC",
"requestParameters" :
```

```
"jobParams.HIST_LOAD_INVRTV_DAY_JOB=2020-09-09 2021-09-09"
}
```

7

Data File Generation

When you are implementing the Retail Analytics Platform without using an Oracle merchandising system for foundation data, or you are providing history data from non-Oracle sources, you will need to create several data files following the platform specifications. This chapter will provide guidance on the data file formats, structures, business rules, and other considerations that must be accounted for when generating the data.

> **Important:**
>
> Do not begin data file creation for RAP until you have reviewed this chapter and have an understanding of the key data structures used throughout the platform.

For complete column-level definitions of the interfaces, including datatype and length requirements, refer to the *RI and Science Interfaces Guide* in My Oracle Support.

Files Types and Data Format

The shared platform data files discussed in this chapter may use a standard comma-delimited (CSV) file format, with text strings optionally enclosed by quotation marks or other characters. The files expect the first line to be column headers, and lines 2+ should contain the input data. This applies to any filename ending in `.csv`. For specific columns in each of these files, the following standards can be used as a guideline (though they can be changed by configuration options).

Table 7-1 Foundation File Formatting

Datatype	Format	Example	Explanation
Number	0.00	340911.10	Numbers should be unformatted with periods for decimal places. Commas or other symbols should not be used within the numerical values.
Character	“abc”	“Item #4561”	Any alphanumeric string can be optionally enclosed by quotation marks to encapsulate special characters such as commas in a descriptive value.
Date	YYYYMMDD	20201231	Dates should be provided as simple 8-digit values with no formatting in year-month-day sequences.

Context File Configuration (CFS Architecture)

> **Note:**
>
> If you are an existing RI or Science customer in our older architecture, or your environments were provisioned initially on version 19 or earlier, then this information does NOT apply. Refer to the section after this for Legacy context file handling.

Before creating and processing a data file on the platform, choose the fields that will be populated and instruct the platform to only look for data in those columns. This configuration is handled through the use of Context (CTX) Files that are uploaded alongside each base data file. For example, the context file for `PRODUCT.csv` will be `PRODUCT.csv.ctx` (appending the `.ctx` file descriptor to the end of the base filename).

Within each context file you must provide a single column containing:

- One or more parameters defining the behavior of the file load and the format of the file.
- The list of fields contained in the source file, in the order in which they appear in the file specification:
  - #TABLE#<Staging Table Name>
  - #DELIMITER#<Input Value>#
  - #DATEFORMAT#<Input Value>#
  - #REJECTLIMIT#<Input Value>#
  - #RECORDDELIMITER#<Input Value>#
  - #IGNOREBLANKLINES#<Input Value>#
  - #SKIPHEADERS#<Input Value>#
  - #TRIMSPACES#<Input Value>#
  - #TRUNCATECOL#<Input Value>#
  - #COLUMNLIST#<Input Value>#

    <COL1>

    <COL2>

    <COL3>

The following is an example context file for the `CALENDAR.csv` data file:

File Name: `CALENDAR.csv.ctx`

File Contents:

```
#TABLE#W_MCAL_PERIOD_DTS#
#DELIMITER#,#
#DATEFORMAT#YYYY-MM-DD#
#REJECTLIMIT#1#
#RECORDDELIMITER#\n#
```

```
#IGNOREBLANKLINES#false#
#SKIPHEADERS#1#
#TRIMSPACES#rtrim#
#TRUNCATECOL#false#
#COLUMNLIST#
MCAL_CAL_ID
MCAL_PERIOD_TYPE
MCAL_PERIOD_NAME
MCAL_PERIOD
MCAL_PERIOD_ST_DT
MCAL_PERIOD_END_DT
MCAL_QTR
MCAL_YEAR
MCAL_QTR_START_DT
MCAL_QTR_END_DT
MCAL_YEAR_START_DT
MCAL_YEAR_END_DT
```

The file must be UNIX formatted and have an end-of-line character on every line, including the last one. As shown above, the final EOL may appear as a new line in a text editor. The `#TABLE#` field is required: it indicates the name of the database staging table updated by the file. The `COLUMNLIST` tag is also required: it determines the columns the customer uses in their `.dat` or `.csv` file. The column list must match the order of fields in the file from left to right, which must also align with the published file specifications. Include the list of columns after the `#COLUMNLIST#` tag. Most of the other parameters are optional and the rows can be excluded from the context file. However, this will set values to system defaults that may not align with your format.

> **Note:**
>
> Both RI and Science can use these context files to determine the format of incoming data.

The server maintains a copy of all the context files used, so you do not need to send a context file every time. If no context files are found, the Analytics Platform uses the last known configuration.

For additional format options, the available values used are from the DBMS_CLOUD package options in ADW.

Legacy Context File Configuration (GBUCS Architecture)

For RI interfaces in environments created from version 19 and earlier (including environments currently on v21 that were provisioned with v19), a more basic CTX context file is used. These context files only accept a list of columns in the data file; they cannot control any formatting or other options. They also must have a fixed filename ending in `.dat.ctx` regardless of the associated file. Use the name of the target table in RI as the file name. For example, `CALENDAR.csv` will have a context file named `W_MCAL_PERIOD_DTS.dat.ctx`. The file format itself is also fixed. CSV files will be comma-delimited, while DAT files will be pipe-delimited. DAT files use the same name for both the data file and context file, such as `W_CODE_DS.dat` and `W_CODE_DS.dat.ctx`.

For a complete list of mappings between CSV filenames and associated table names used for CTX file naming, refer to Context File Table Reference .

The following is an example context file for the `CALENDAR.csv` data file:

File Name: `W_MCAL_PERIOD_DTS.dat.ctx`

File Contents:

```
MCAL_CAL_ID
MCAL_PERIOD_TYPE
MCAL_PERIOD_NAME
MCAL_PERIOD
MCAL_PERIOD_ST_DT
MCAL_PERIOD_END_DT
MCAL_QTR
MCAL_YEAR
MCAL_QTR_START_DT
MCAL_QTR_END_DT
MCAL_YEAR_START_DT
MCAL_YEAR_END_DT
```

> **Note:**
>
> This legacy context file format applies strictly to Retail Insights. Science applications do not support the legacy context file format; they expect the full data file contents in their interfaces.

Application-Specific Data Formats

Each application within the Retail Analytics Platform may require data to be provided using specific rules and data formats, which can differ from those used in the common platform files. This section describes the use-cases for alternate data formats and lays out the basic rules that must be followed.

Retail Insights

Retail Insights has a large number of legacy interfaces that do not follow the shared platform data formats. These interfaces are populated with files named after their target database table with a file extension of `.dat`, such as `W_PRODUCT_DS.dat`. All files ending with a `.dat` extension are pipe-delimited files (using the `|` symbol as the column separator). These files also have a Unix line-ending character by default, although the line-ending character can be configured to be a different value, if needed. These files may be created by a legacy Merchandising (RMS) extract process or may be produced through existing integrations to an older version of RI or Science.

Table 7-2 Retail Insights Legacy File Formatting

Datatype	Format	Example	Explanation
Number	0.00	340911.10	Unformatted numbers with periods for decimal places. Commas or other symbols cannot be used within the numerical values.
Character	abc	Item #4561	Any alphanumeric string will provided as-is, with the exception that it must NOT contain pipe characters or line-ending characters.
Date	YYYY-MM-DD;00:00:00	2020-05-09;00:00:00	Dates without timestamps must still use a timestamp format, but they must be hard-coded to have a time of 00:00:00. Date fields (such as `DAY_DT` columns) must NOT have a non-zero time, or they will not load correctly.
Timestamp	YYYY-MM-DD;HH:MM:SS	2020-05-09;09:35:19	Use full date-and-time formatting ONLY when a full timestamp is expected on the column. This is not commonly used and should be noted in the interface specifications, if supported.

If you are implementing Retail Insights as one of your modules and you are in an environment that was originally installed with version 19 or earlier of RI, you may need to provide some files in this data format, in addition to the foundation files which use the CSV format. This file format is also used when integrating with legacy solutions such as the Retail Merchandising System (RMS) through the Retail Data Extractor (RDE).

Example data from the file `W_RTL_PLAN1_PROD1_LC1_T1_FS.dat`:

```
70|-1|13|-1|2019-05-04;00:00:00|RETAIL|0|1118.82|1|
70~13~2019-05-04;00:00:00~0
70|-1|13|-1|2019-05-11;00:00:00|RETAIL|0|476.09|1|70~13~2019-05-11;00:00:00~0
70|-1|13|-1|2019-05-18;00:00:00|RETAIL|0|296.62|1|70~13~2019-05-18;00:00:00~0
```

Retail Science Platform

Modules within the Science Platform leverage the same Context (CTX) file concepts as described in the common foundation file formats. You may control the structure and contents of Science files using the parameters in the context files. The full list of interfaces used by Science modules is included in the *Interfaces Guide*.

Planning Platform

Planning solutions using PDS (Planning Data Schema), such as Merchandise Financial Planning, have two main types of files:

Hierarchy/Dimension Files – Foundation Data for the Hierarchy/Dimensions.

Measure/Fact Files – Factual Data specific to loadable metric/measures.

When loading directly to Planning applications, both types of files should only be in CSV format and they should contain headers. Headers contain the details of the dimension names for Hierarchy/Dimension Files and the fact names for Measure/Fact Files.

Hierarchy/Dimension Files uses the naming convention `<Hierarchy Name>.hdr.csv.dat` and Measure Files can be any meaningful fact-grouping name, but with allowed extensions such as `.ovr`, `.rpl`, or `.inc`.

- OVR extension is used for override files
- RPL extension is used to delete and replace position-based data sets
- INC extension is for incremental files that can increment positional data.

If using the common foundation CSV files, most of the data can be interfaced using those shared integrations. However, certain files (such as the VAT Hierarchy) must be directly loaded to Planning: it does not come from RI at this time. Refer to the application-specific Planning Implementation Guides for more details about the list of files that are not included in foundation integrations.

Dimension Files

A dimension is a collection of descriptive elements, attributes, or hierarchical structures that provide context to your business data. Dimensions tell the platform what your business looks like and how it operates. They describe the factual data (such as sales transactions) and provide means for aggregation and summarization throughout the platform. Dimensions follow a strict set of business rules and formatting requirements that must be followed when generating the files.

There are certain common rules that apply across all of the dimension files and must be followed without exception. Failure to adhere to these rules may result in failed data loads or incorrectly structured datasets in the platform.

- All dimension files must be provided as full snapshots of the source data at all times, unless you change the configuration of a specific dimension to be `IS_INCREMENTAL=Y` where incremental loads are supported. Incremental dimension loading should only be done once nightly/weekly batch processing has started. Initial/history dimension loads should always be full snapshots.
- Hierarchy levels must follow a strict tree structure, where each parent has a 1-to-N relationship with the children elements below them. You cannot have the same child level identifier repeat across more than one parent level, with the exception of Class/Subclass levels (which may repeat on the ID columns but must be unique on the UID columns). For example, Department 12 can only exist under Division 1, it cannot also exist under Division 2.
- Hierarchy files (product, organization, calendar) must have a value in all non-null fields for all rows and must fill in all the required hierarchy levels without exception. For example, even if your non-Oracle product data only has 4 hierarchy levels, you must provide the complete 7-level product hierarchy to the platform. Fill in the upper levels of the hierarchy with values to make up for the differences, such as having the division and group levels both be a single, hard-coded value.
- Any time you are providing a key identifier of an entity (such as a supplier ID, channel ID, brand ID, and so on) you should fill in the values on all rows of the data file, using a dummy value for rows that don’t have that entity. For example, for items that don’t have a brand, you can assign them to a generic “No Brand” value

to support filtering and reporting on these records throughout the platform. You may find it easier to identify the “No Brand” group of products when working with CDTs in the Science platform or when creating dashboards in RI, compared to leaving the values empty in the file.

- Any change to hierarchy levels after the first dimension is loaded will be treated as a reclassification and will have certain internal processes and data changes triggered as a result. If possible, avoid loading hierarchy changes to levels above Item/Location during the historical load process. If you need to load new hierarchies during the history loads, make sure to advance the business date in RI using the specified jobs and date parameters, do NOT load altered hierarchies on top of the same business date as previous loads.
- All fields designated as flags (having FLG or FLAG in the field name) must have a `Y` or `N` value. Filters and analytics within the system will generally assume `Y`/`N` is used and not function properly if other values (like `0`/`1`) are provided.
- Retail Insights requires that all hierarchy identifiers above item/location level MUST be numerical. The reporting layer is designed around having numerical identifiers in hierarchies and no data will show in reports if that is not followed. If you are not implementing Retail Insights, then alphanumeric hierarchy IDs could be used, though it is not preferred.

Product File

The product file is named `PRODUCT.csv`, and it contains most of the identifying information about the merchandise you sell and the services you provide. The file structure follows certain rules based on the Retail Merchandising Foundation Cloud Services (RMFCS) data model, as that is the paradigm for retail foundation data that we are following across all RAP foundation files.

The columns below are the minimum required data elements, but the file supports many more optional fields, as listed in the Interfaces Guide. Optional fields tend to be used as reporting attributes in RI and are nullable descriptive fields. Optional fields designated for use in a Science or Planning module are generally nullable too, but should generally be populated with non-null values to provide more complete data to those modules.

Table 7-3 Product File Required Fields

Column Header	Usage
ITEM	Product number which uniquely identifies the record. Could be any of these records: • a sub-transaction level item (such as a UPC) which has a SKU as a parent and Style as a grandparent • a transaction-level SKU (with or without parents) • a style or "level 1" item which is above transaction level
ITEM_PARENT	Parent item associated with this record when the item level is `2` or `3`. REQUIRED when providing multiple levels of items to establish the parent/child relationships.
ITEM_GRANDPARENT	Grandparent item associated with this record when the item level is `3`. REQUIRED when providing multiple levels of items to establish the parent/child relationships.

Table 7-3 (Cont.) Product File Required Fields

Column Header	Usage
ITEM_LEVEL	Item Level (1, 2, or 3) of this record from the source system. Used to determine what level of item is being provided to the target systems. Item level 2 should have a parent item, and item level 3 should provide both parent and grandparent. Typical fashion item levels are: • Level 1 – Style • Level 2 – SKU (transaction level) • Level 3 – UPC/EAN/barcode
TRAN_LEVEL	Transaction level (1 or 2) of the item from the source system. Identifies which level is used as the transaction level in RI and RSP.
PACK_FLG	Pack flag (where N = regular item, Y = pack item). REQUIRED column if the retailer has packs, Optional otherwise.
DIFF_AGGREGATE	Combined differentiator values are used in defining the diff aggregate level (in between Item level 1 and 2 for a multi-level item). For example, for a fashion item, this will be the Color. Specify this on the transaction item-level records.
LVL4_PRODCAT_ID	Default level for Subclass, which is the first level above item in the hierarchy structure. Sometimes referred to as segment or subsegment. All items of any type are mapped to a subclass as their first level. Parent items are not to be treated as hierarchy levels in the file.
LVL4_PRODCAT_UID	Unique identifier of the Subclass. In many merchandising systems the subclass is not unique on its own, so a separate, unique key value must be provided in this case.
LVL5_PRODCAT_ID	Default level for Class (sometimes referred to as Subcategory).
LVL5_PRODCAT_UID	Unique identifier of the Class. In many merchandising systems the class is not unique on its own, so a separate unique key value must be provided in this case.
LVL6_PRODCAT_ID	Default Level for Department (also referred to as Category).
LVL7_PRODCAT_ID	Default Level for Group.
LVL8_PRODCAT_ID	Default Level for Division.
TOP_PRODCAT_ID	Default Level for Company. Only one company is supported at this time, you may not have 2+ companies in the same instance.
ITEM_DESC	Product Name or primary item description.
LVL4_PRODCAT_DESC	Default Level for Subclass Description.
LVL5_PRODCAT_DESC	Default Level for Class Description.
LVL6_PRODCAT_DESC	Default Level for Department Description.
LVL7_PRODCAT_DESC	Default Level for Group Description.
LVL8_PRODCAT_DESC	Default Level for Division Description.
TOP_PRODCAT_DESC	Default Level for Company Description.

The product hierarchy fields use generic level names to support non-traditional hierarchy structures (for example, your first hierarchy level may not be called Subclass, but you are still loading it into the same position in the file). Other levels

such as LVL1 to LVL3 have columns in the interface but are not yet used in any module of the platform.

> **Note:**
>
> Multi-level items are not always required and depend on your use-cases. For example, the lowest level (`ITEM_LEVEL=3`) for sub-transaction items is only used in Retail Insights for reporting on UPC or barcode level attribute values. If you are a non-fashion retailer you may only have a single item level (for SKUs) and the other levels could be ignored. The reason for having different records for each item level is to allow for different attributes at each level, which can be very important in Retail Insights analytics. You may also need to provide multiple item levels for optimizing or planning data at a Style or Style/Color level in the non-RI modules. When providing multiple item level records, note that the item IDs must be unique across all levels and records.

Example data for the `PRODUCT.csv` file columns above, including all 3 supported item levels:

```
ITEM,ITEM_PARENT,ITEM_GRANDPARENT,ITEM_LEVEL,TRAN_LEVEL,PACK_FLG,DIFF_AGGREGA
TE,LVL4_PRODCAT_ID,LVL4_PRODCAT_UID,LVL5_PRODCAT_ID,LVL5_PRODCAT_UID,LVL6_PRO
DCAT_ID,LVL7_PRODCAT_ID,LVL8_PRODCAT_ID,TOP_LVL_PRODCAT_ID,ITEM_DESC,LVL4_PRO
DCAT_DESC,LVL5_PRODCAT_DESC,LVL6_PRODCAT_DESC,LVL7_PRODCAT_DESC,LVL8_PRODCAT_
DESC,TOP_LVL_PRODCAT_DESC
190085210200,-1,-1,1,2,N,,8,9001,3,910,3,2,1,1,2IN1 SHORTS,Shorts,Active
Apparel,Women's Activewear,Activewear,Apparel,Retailer Ltd
190085205725,190085210200,-1,2,2,N,BLK,8,9001,3,910,3,2,1,1,2IN1
SHORTS:BLACK:LARGE,Shorts,Active Apparel,Women's
Activewear,Activewear,Apparel,Retailer Ltd
190085205923,190085210200,-1,2,2,N,DG,8,9001,3,910,3,2,1,1,2IN1 SHORTS:DARK
GREY:LARGE,Shorts,Active Apparel,Women's
Activewear,Activewear,Apparel,Retailer Ltd
1190085205725,190085205725,190085210200,3,2,N,,8,9001,3,910,3,2,1,1,2IN1
SHORTS:BLACK:LARGE:BC,Shorts,Active Apparel,Women's
Activewear,Activewear,Apparel,Retailer Ltd
1190085205923,190085205923,190085210200,3,2,N,,8,9001,3,910,3,2,1,1,2IN1
SHORTS:DARK GREY:LARGE:BC,Shorts,Active Apparel,Women's
Activewear,Activewear,Apparel,Retailer Ltd
```

For other fields not shown here, they are optional from a data load perspective but may be used by one or more applications on the platform, so it is best to consider all fields on the interface and populate as much data as you can. For example, supplier information is a requirement for Inventory Optimization, and brand information is often used in Clustering or Demand Transference. Also note that some fields come in pairs and must be provided together or not at all. This includes:

- Brand name and description
- Supplier ID and description

Description fields could be set to the same value as the identifier if no other value is known or used, but you must include both fields with non-null values when you want to provide the data.

Organization File

The organization file will contain most of the identifying information about the locations where you sell or store merchandise, including physical locations (such as a brick & mortar store) and virtual locations (such as a web store or virtual warehouse entity). The file structure follows certain rules based on the Retail Merchandising Foundation Cloud Services (RMFCS) data model, as that is the paradigm for retail foundation data that we are following across all RAP foundation files. The columns below are the minimum required data elements, but the file supports many more optional fields, as listed in the Interfaces Guide.

Table 7-4 Organization File Required Fields

Column Header	Usage
ORG_NUM	The external identifier for a location, including stores, warehouses, and partner locations.
ORG_TYPE_CODE	The type code of the location. It must be one of `S`, `W`, or `E`, representing a Store, Warehouse, or External location.
CURR_CODE	This is the 3-character base currency code of the organizational entity, such as `USD` or `CAN`.
ORG_HIER10_NUM	Default Level for District, which is the first level above Location in the hierarchy.
ORG_HIER11_NUM	Default Level for Region.
ORG_HIER12_NUM	Default Level for Area.
ORG_HIER13_NUM	Default Level for Chain.
ORG_TOP_NUM	Default Level for Company. Only one company is supported at this time. You may not have 2+ companies in the same instance.
ORG_DESC	Short name or short description of the location.
ORG_SECONDARY_DESC	Full name or long description of the location.
ORG_HIER10_DESC	Default Level for District Description.
ORG_HIER11_DESC	Default Level for Region Description.
ORG_HIER12_DESC	Default Level for Area Description.
ORG_HIER13_DESC	Default Level for Chain Description.
ORG_TOP_DESC	Default Level for Company Description.

The organization hierarchy fields use generic level names to support non-traditional hierarchy levels (for example, your first hierarchy level may not be called District, but you are still loading it into the same position in the file which is used for Districts). Other levels, such as `1` to `9`, have columns in the interface but are not yet used in any module of the platform.

> **Note:**
>
> Warehouses may not have a hierarchy structure in your core organizational hierarchy. If that is the case, you should provide a separate reserved value like `1` on all hierarchy level numbers between location and company. This will prevent any potential roll-up issues, such as warehouse inventory being included in Science optimization runs at levels above location.

Example data for the `ORGANIZATION.csv` file columns above as well as some optional fields available on the interface:

```
ORG_NUM,ORG_TYPE_CODE,CURR_CODE,STATE_PROV_NAME,COUNTRY_REGION_NAME,ORG_HIER1
0_NUM,ORG_HIER11_NUM,ORG_HIER12_NUM,ORG_HIER13_NUM,ORG_TOP_NUM,ORG_DESC,ORG_S
ECONDARY_DESC,ORG_HIER10_DESC,ORG_HIER11_DESC,ORG_HIER12_DESC,ORG_HIER13_DESC
,ORG_TOP_DESC,CHANNEL_ID,CHANNEL_NAME,PHYS_WH_ID,STOCKHOLDING_FLG,STORE_FORMA
T_DESC,STORE_FORMAT_ID,STORE_TYPE,TRANSFER_ZONE_ID,TRANSFER_ZONE_DESC,VIRTUAL
_WH_FLG,STORE_CLASS_TYPE,STORE_CLASS_DESC,WH_DELIVERY_POLICY,WH_REPL_IND,DUNS
_NUMBER,STORE_REMODEL_DT,STORE_CLOSE_DT,INBOUND_HANDLING_DAYS,FLEX1_CHAR_VALU
E,FLEX2_CHAR_VALUE,FLEX3_CHAR_VALUE,FLEX4_CHAR_VALUE,FLEX5_CHAR_VALUE,FLEX6_C
HAR_VALUE,FLEX7_CHAR_VALUE,FLEX8_CHAR_VALUE,FLEX9_CHAR_VALUE,FLEX10_CHAR_VALU
E
1000,S,USD,North Carolina,United
States,1070,170,1,1,1,Charlotte,Charlotte,North Carolina,Mid-Atlantic,Brick
& Mortar,US,Retailer Ltd,1,North America,,Y,Store,1,C,101,Zone
101,N,1,A,,,,,,,WH-1,Warehouse - US,1,Store Pick Up / Take
With,3,Comp,6,Mixed Humid,1,Very Large
1001,S,USD,Georgia,United
States,1023,400,1,1,1,Atlanta,Atlanta,Georgia,South Atlantic,Brick &
Mortar,US,Retailer Ltd,1,North America,,Y,Kiosk,2,C,101,Zone
101,N,6,F,,,,,,,WH-1,Warehouse - US,2,Deliver/Install at
Customer ,3,Comp,7,Hot Humid,3,Medium
1002,S,USD,Texas,United States,1104,230,1,1,1,Dallas,Dallas,Texas,Gulf
States,Brick & Mortar,US,Retailer Ltd,1,North America,,Y,Store,1,C,101,Zone
101,N,6,F,,,,,,,WH-1,Warehouse - US,3,Home Delivery,3,Comp,4,Hot Dry,3,Medium
```

Also note that some optional fields come in pairs and must be provided together or not at all. This includes:

- Banner ID and description
- Channel ID and description
- Store format ID and description

Description fields can be set to the same value as the identifier if no other value is known or used, but you must include both fields with non-null values when you provide the data.

Calendar File

The calendar file contains your primary business or fiscal calendar, defined at the fiscal-period level of detail. The most common fiscal calendar used is a 4-5-4 National Retail Federation (NRF) calendar or a variation of it with different year-ending dates. This calendar defines the financial, analytical, or planning periods used by the business. It must contain some form of fiscal calendar, but if you are a business that operates solely on the Gregorian

calendar, a default calendar file can be generated by an ad hoc batch program to initialize the system. However, if you are implementing a planning solution, you must use the Fiscal Calendar as your primary calendar, and only this calendar will be integrated from RI to Planning.

Table 7-5 Calendar File Required Fields

Column Header	Usage
MCAL_CAL_ID	Identifies the accounting calendar. At this time a hard-coded value of `Retail Calendar~41` is expected here, no other value should be used.
MCAL_PERIOD_TYPE	Identifies the accounting period type (`4` or `5`). This represents if the fiscal period has 4 or 5 weeks.
MCAL_PERIOD_NAME	Name of the fiscal period, such as `Period01`, `Period02`, and so on. Do not include the year in this name, as the system will automatically add that during the load.
MCAL_PERIOD	Period number within a year, for a 4-5-4 calendar this should be between `1` and `12`.
MCAL_PERIOD_ST_DT	Identifies the first date of the period in YYYYMMDD format (default format can be changed in CTX files).
MCAL_PERIOD_END_DT	Identifies the last date of the period in YYYYMMDD format (default format can be changed in CTX files).
MCAL_QTR	Identifies the quarter of the year to which this period belongs. Possible values are `1`, `2`, `3` and `4`.
MCAL_YEAR	Identifies the fiscal year in YYYY format.
MCAL_QTR_START_DT	Identifies the start date of the quarter in YYYYMMDD format (default format can be changed in CTX files).
MCAL_QTR_END_DT	Identifies the end date of the quarter in YYYYMMDD format (default format can be changed in CTX files).
MCAL_YEAR_START_DT	Identifies the start date of the year in YYYYMMDD format (default format can be changed in CTX files).
MCAL_YEAR_END_DT	Identifies the end date of the year in YYYYMMDD format (default format can be changed in CTX files).

The hard-coded calendar ID is used to align with several internal tables that are designed to support multiple calendars but currently have only one in place, and that calendar uses the provided value of `MCAL_CAL_ID` above.

The fiscal calendar should have, at a minimum, a 5-year range (2 years in the past, the current fiscal year, and 2 years forward from that) but is usually much longer so that you do not need to update the file often. Most implementations should start with a 10-15 year fiscal calendar length. The calendar should start at least 1 full year before the planned beginning of your history files and extend at least 1 year beyond your expected business needs in all RAP modules.

Example data for the `CALENDAR.csv` file columns above:

```
MCAL_CAL_ID,MCAL_PERIOD_TYPE,MCAL_PERIOD_NAME,MCAL_PERIOD,MCAL_PERIOD_S
T_DT,MCAL_PERIOD_END_DT,MCAL_QTR,MCAL_YEAR,MCAL_QTR_START_DT,MCAL_QTR_E
ND_DT,MCAL_YEAR_START_DT,MCAL_YEAR_END_DT
Retail
```

```
Calendar~41,4,Period01,1,20070204,20070303,1,2007,20070204,20070505,20070204,
20080202
Retail
Calendar~41,5,Period02,2,20070304,20070407,1,2007,20070204,20070505,20070204,
20080202
Retail
Calendar~41,4,Period03,3,20070408,20070505,1,2007,20070204,20070505,20070204,
20080202
Retail
Calendar~41,4,Period04,4,20070506,20070602,2,2007,20070506,20070804,20070204,
20080202
```

Exchange Rates File

The exchange rates file captures conversion rates between any two currency codes that may appear in your fact data. The standard practice for fact data is to load the source system values in the original currency and allow the platform to convert the amounts to the primary currency. This file facilitates that process and triggers bulk updates in nightly processing any time you wish to change your exchange rates for financial reporting purposes. Adding new rates to the file with an effective date equal to the batch date triggers a mass update to all positional facts, converting all the amounts to the new exchange rate even if the item/location did not otherwise change in the source system.

Note that you do not have to provide exchange rates data for the following scenarios:

- You are loading your data already converted to primary currency
- You only use a single currency for your entire business
- You are only implementing a Science module and expect to perform those processes in the local currency amounts

Even when exchange rates are not required as a separate file, you should still populate the default currency codes (`DOC_CURR_CODE`, `LOC_CURR_CODE`, `LOC_EXCHANGE_RATE`) in the fact data files with default values.

Table 7-6 Exchange Rate File Required Fields

Column Header	Usage
START_DT	Contains the effective start date of the exchange rate. Set to the current business date to trigger new rate conversions.
END_DT	Contains the effective end date of the exchange rate. Default to `21000101` if the rate should be effective indefinitely.
EXCHANGE_RATE	Contains the exchange rate for the specified currency/type/effective date combination.
FROM_CURRENCY_CODE	Code of the currency to be exchanged.
TO_CURRENCY_CODE	Code of the currency to which a currency is exchanged.

Sample data for the EXCH_RATE.csv file columns above:

```
START_DT,END_DT,EXCHANGE_RATE,FROM_CURRENCY_CODE,TO_CURRENCY_CODE
20180514,21000101,0.8640055,CAD,USD
```

```
20180514,21000101,0.1233959,CNY,USD
20180514,21000101,1,USD,USD
```

The exchange rates data must also satisfy the following criteria if you are loading data for use in Retail Insights reporting:

1. Rates must be provided in both directions for every combination of currencies that can occur in your dataset (for example, USD > CAD and CAD > USD).
2. Rows must be included for each currency to itself with a fixed rate of 1 and end date of 2100-01-01 (as shown in the above example). These rows are used when RI converts data to document (source) currencies, to display the correct amounts on rows that have the same local and primary currency.
3. Dates must provide complete coverage of your entire timeframe in the dataset, both for historical and current data. The current effective records for all rates can use 2100-01-01 as the end date. Dates cannot overlap, only a single rate must be effective per day.
4. Rates should not change more often than absolutely necessary based on the business requirements. If you are implementing RI with positional data, a rate change triggers a complete recalculation of the stock on hand cost/retail amounts for the entire business across all pre-calculated aggregate tables. When RI is not used for financial reporting you might only change the rates once each fiscal year, to maintain a single constant currency for analytical purposes.

Attributes Files

Product attributes are provided on two files: one file for the attribute-to-product mappings and another for attribute descriptions and codes. These files should be provided together to fully describe all the attributes being loaded into the system. The attribute descriptors file must be a full snapshot of all attribute types and values at all times. The product attribute mapping file should start as a full snapshot but can move to incremental (delta) load methods once nightly batches begin, if you can extract the information as deltas only.

Product attributes are a major component of the RI and Science modules and drive many analytical processes but are not required for some planning modules like MFP.

Table 7-7 Attribute File Required Fields

Column Header	Usage
ATTR_VALUE_ID	Unique identifier for a user-defined attribute or product differentiator. This interface contains ALL values regardless of whether they are used on items yet or not. These values must also match the `ATTR_ID` on the other file.
ATTR_VALUE_DESC	Descriptive value for a user-defined attribute or product differentiator, such as `Brown`, `Cotton`, `Size12`, and so on
ATTR_GROUP_ID	Unique identifier for a group of user-defined attributes, or the name/code for the differentiator group. Reserved attribute type codes like `SIZE` must all have a single value associated with the group in this field, such as `S` or `SZ`. The value can be hard-coded if you do not have a code representing that group in your source system. This value must also match the `ATTR_GRP_ID` value on the other file.

Table 7-7 (Cont.) Attribute File Required Fields

Column Header	Usage
ATTR_GROUP_DESC	Descriptive value for a group of user-defined attributes, such as `Color Family` or `Web Exclusive Code`, or the description of the differentiator type such as `Color`.
ATTR_TYPE_CODE	Indicates the type of UDA or differentiator. UDA types should be hard-coded as one of `FF`, `LV`, or `DT`, for Freeform, List of Values, or Date. `LV` type attributes are fixed lists and write values to translation lookup tables, while `FF` and `DT` fields do not. Some diffs have special reserved codes for this field as well, which should be used when applicable, and include `SIZE`, `FABRIC`, `SCENT`, `FLAVOR`, `STYLE`, and `COLOR`. Other differentiators not using these codes should specify a hard-coded type of `DIFF`.

Sample data for the `ATTR.csv` file columns above:

```
ATTR_VALUE_ID,ATTR_VALUE_DESC,ATTR_GROUP_ID,ATTR_GROUP_DESC,ATTR_TYPE_CODE
13,No_Sugar_IN13,45008,UDA_ING_2018.01.16.01.00,FF
14,Zero_Carbs_IN14,45008,UDA_ING_2018.01.16.01.00,FF
3,Distressed,80008,Wash,LV
CHOC,Chocolate,FLAVOR,Flavor,FLAVOR
GRAY_43214,Gray,C,Color,COLOR
32X32_9957,32X32,S,Size,SIZE
```

Table 7-8 Product Attribute File Required Fields

Column Header	Usage
ITEM	The item number associated with the specified attribute value.
ATTR_ID	Identifier for an attribute value, such as the UDA value ID or Diff ID which is mapped to the item on the record, or the ID for the item list this item belongs to. These values must also match the `ATTR_VALUE_ID` on the other file.
ATTR_GRP_TYPE	The attribute group type. This is a set of fixed values which must be selected from what RAP supports. Supported values are `ITEMDIFF`, `ITEMUDA`, `ITEMLIST`, `COLOR`, and `PRODUCT_ATTRIBUTES`. These codes determine the target columns for the data (for example, lists, diffs, and UDAs use different internal columns in the data model). `PRODUCT_ATTRIBUTES` type code encapsulates the other named differentiator types like Size, Fabric, and so on. `COLOR` has a special type code due to it being a common level between Style and SKU for fashion retailers, so it is handled separately.

Table 7-8 (Cont.) Product Attribute File Required Fields

Column Header	Usage
ATTR_GRP_ID	Identifier for the attribute group containing the value on this record. Varies by ATTR_GRP_TYPE value used: • ITEMDIFF, COLOR, PRODUCT_ATTRIBUTES = specify the Diff Type ID, equivalent to an attribute group ID for diffs (align to ATTR.csv group ID) • ITEMUDA = UDA Group ID (align to ATTR.csv group ID) • ITEMLIST = <N/A - Not Used for Lists, leave null>
DIFF_GRP_ID	Differentiator group used to assign the diff attribute on this item; for example the Size Group ID used when generating SKUs from a parent Style. Only SKUs will have diff groups associated with them. Only diffs will have groups, not UDAs or other record types. This is not the same as an attribute group, which is the overall grouping of attribute values across all items. This is used mainly for Size Profile Science. Foreign key reference to DIFF_GROUP.csv.
DIFF_GRP_DESC	Descriptive value of the diff group mapped to this item/attribute record.

Sample data for the PROD_ATTR.csv file columns above:

```
ITEM,ATTR_ID,ATTR_GRP_TYPE,ATTR_GRP_ID,DIFF_GRP_ID,DIFF_GRP_DESC
91207973747,30007,ITEMUDA,5,,
91207973747,12756,ITEMUDA,3,,
190496585706,STEEL,ITEMDIFF,METAL,,
86323133004,GRAY_43214,COLOR,C,,
190085302141,CHOC,PRODUCT_ATTRIBUTES,FLAVOR,,
345873291,32X32_9957,PRODUCT_ATTRIBUTES,S,S13,Pant Sizes
```

Fact Files

Fact Data Key Columns

Fact data files, such as sales and inventory, share many common characteristics and standards that can be followed regardless of the specific file. The table below summarizes those key elements and their minimum requirements. You will generally always want to populate these fields where they exist on an interface file.

Table 7-9 Common Fact Data Fields

Column Header	Usage
ITEM	Unique identifier of a transaction item. Must align with the product records in the PRODUCT.csv file where ITEM_LEVEL = TRAN_LEVEL.
ORG_NUM	Unique identifier of an organizational entity. Must align with the location records in the ORGANIZATION.csv file.

Table 7-9 (Cont.) Common Fact Data Fields

Column Header	Usage
DAY_DT	Transaction date or business date for the fact data entry, formatted as YYYYMMDD. Must be a valid date within the range of periods in CALENDER.csv.
RTL_TYPE_CODE	Retail types define a general category for the record that varies by interface. For Sales and Markdowns, it must be one of R/P/C, representing the regular/promo/clearance status of the transaction.
*_QTY	Fields ending with QTY represent the quantity or units of an item on the record, such as the units sold on a transaction line or the units of on-hand inventory.
*_AMT_LCL	Fields containing AMT_LCL represent the currency amount of a record in the local currency of the source system, such as sales retail amount in Canadian dollars from a Canada location, or the cost value of on-hand inventory at your U.S. warehouse.
DOC_CURR_CODE	The original document currency of the record in the source system. For example, a sale made in Canada may have a value of CAD in this field.
LOC_CURR_CODE	The local operating currency of your main office or headquarters. A company based in the United States would use USD in this field.

Nearly all fact files share a common intersection of an item, location, and date as specified above. Such files are expected to come into the platform on a nightly basis and contain that day's transactions or business activity.

Most fact data also supports having currency amounts in their source currency, which is then automatically converted to your primary operating currency during the load process. There are several currency code and exchange rate columns on such interfaces, which should be populated if you need this functionality. The most important ones are shown in the list above, and other optional column for global currencies can be found in the *Interfaces Guide*. When you provide these fields, they must all be provided on every row of data, you cannot leave out any of the values or it will not load properly.

In addition to the commonly used fields, most files have other key columns that can be used to specify important details about the record. Some of these fields are marked as optional in the full interface specifications, meaning that if they are not provided, then some default values will be set for you.

Table 7-10 Additional Fact Data Key Columns

File	Column Header	Usage
SALES	SLS_TRX_ID	Unique identifier of a sales transaction. By default, it is expected sales data will come at the most granular level (transaction-line). If you are not able to provide transaction-level data, specify another unique value in this field.

Table 7-10 (Cont.) Additional Fact Data Key Columns

File	Column Header	Usage
SALES	MIN_NUM	Hour and minute of the transaction in 24-hour, 4-digit format; for example, 11:23 PM would be 2323. Currently only used in Retail Insights reporting; default to `0` if not needed.
SALES	IT_SEQ_NUM	Sequence of a line in the transaction. Every line of a transaction must have its own sequence number. This facilitates uniqueness requirements where the same item could have multiple lines. Without a unique sequence number, the line would not be unique on the input file and the system would see them as duplicate records (and they will not load properly as a result).
SALES	PROMO_ID PROMO_COMP_ID	This two-part identifier maps to the Promotion and Offer associated with a transaction line. They are required if you wish to load sales by promotion for Retail Insights or certain Science modules such as Demand Transference.
SALES	CASHIER_ID SALES_PERSON_ID	Cashier or salesperson associated with the transaction. Currently used only for Retail Insights.
SALES	CO_HEAD_ID CO_LINE_ID	Order ID and line ID for a customer order, such as an online sale. Currently only used by Retail Insights.
SALES	CUSTOMER_NUM CUSTOMER_TYPE	Customer identifier on a sales transaction, which is a critical piece of information for many Science modules, such as Customer Segmentation and Consumer Decisions Trees. `CUSTOMER_TYPE` should not be used at this time, it serves no purpose in downstream modules. If you do include the column, hard-code it to a value of `CUSTID`, or `CUSTOMER_NUM` will be ignored.
SALES_PACK	ITEM PACK_NUM	`ITEM` on the sales pack interface is the component item inside a selling pack. The `PACK_NUM` is the identifier for the pack sold to the customer. The interface itself should be pro-rated component level sales spread down from the selling pack level. Only component sales are used in Planning.
INVENTORY	CLEARANCE_FLG	`Y/N` indicator for whether the item/location is currently on clearance.

Table 7-10 (Cont.) Additional Fact Data Key Columns

File	Column Header	Usage
RECEIPT	INVRC_TYPE_CODE	Indicates the type of receipt into a location using merchandising transaction codes `20` (purchase order receipt), `44~A` (allocation transfer receipt), or `44~T` (non-allocation transfer receipt). Only PO receipts are sent to planning applications, as they represent actual inventory entering the company, and not just movement between two owned locations.
ADJUSTMENT	REASON_CODE	User-defined reason for making the inventory adjustment. Currently used only for Retail Insights.
ADJUSTMENT	INVADJ_TYPE_CODE	Indicates the type of adjustment to inventory using merchandise transaction codes `22` (non-shrink adjustment), `23` (shrink adjustment), or `41` (stock count adjustment). The codes determine the associated Planning measures the data is placed into.
TRANSFER	FROM_ORG_NUM FROM_CLEARANCE_FLG FROM_*	Transfers are created for both the source of the transfer (the FROM location) and the destination (the TO location). Fields not having a `FROM_` prefix specify the destination of the transfer.
TRANSFER	TSF_TYPE_ID	The type of transfer in the source system. It must be `N`, `B`, or `I`, representing a normal, book, or intercompany transfer.
RTV	INV_STATUS	Status codes for the type of inventory being returned to a vendor, such as damaged or unsellable.

Here are sample records for commonly used historical load files having a small set of fields populated. These fields are sufficient to see results in RI reporting and move the data to RSP or MFP but may not satisfy all the functional requirements of those applications. Review the interfaces guide for complete details on required/optional columns on these interfaces.

`SALES.csv`:

```
ITEM,ORG_NUM,DAY_DT,MIN_NUM,RTL_TYPE_CODE,SLS_TRX_ID,PROMO_ID,PROMO_COMP_ID,C
ASHIER_ID,REGISTER_ID,SALES_PERSON_ID,CUSTOMER_NUM,SLS_QTY,SLS_AMT_LCL,SLS_PR
OFIT_AMT_LCL,RET_QTY,RET_AMT_LCL,RET_PROFIT_AMT_LCL,TRAN_TYPE,LOC_CURR_CODE,D
OC_CURR_CODE
1235842,1029,20210228,0,R,202102281029,-1,-1,96,19,65,-1,173,1730,605.5,0,0,0
,SALE,USD,USD
1235842,1029,20210307,0,R,202103071029,-1,-1,12,19,55,-1,167,1670,584.5,0,0,0
,SALE,USD,USD
1235842,1029,20210314,0,R,202103141029,-1,-1,30,18,20,-1,181,1810,633.5,0,0,0
,SALE,USD,USD
```

INVENTORY.csv:

```
ITEM,ORG_NUM,DAY_DT,CLEARANCE_FLG,INV_SOH_QTY,INV_SOH_COST_AMT_LCL,INV_
SOH_RTL_AMT_LCL,INV_UNIT_RTL_AMT_LCL,INV_AVG_COST_AMT_LCL,INV_UNIT_COST
_AMT_LCL,PURCH_TYPE_CODE,DOC_CURR_CODE,LOC_CURR_CODE
72939751,1001,20200208,N,0,0,0,104.63,0,48.52,0,USD,USD
73137693,1001,20200208,N,0,0,0,104.63,0,48.52,0,USD,USD
75539075,1001,20200208,N,0,0,0,101.73,0,47.44,0,USD,USD
```

PRICE.csv:

```
ITEM,ORG_NUM,DAY_DT,PRICE_CHANGE_TRAN_TYPE,SELLING_UOM,STANDARD_UNIT_RT
L_AMT_LCL,SELLING_UNIT_RTL_AMT_LCL,BASE_COST_AMT_LCL,LOC_CURR_CODE,DOC_
CURR_CODE
89833651,1004,20200208,0,EA,93.11,93.11,53.56,USD,USD
90710567,1004,20200208,0,EA,90.41,90.41,50.74,USD,USD
90846443,1004,20200208,0,EA,79.87,79.87,44.57,USD,USD
```

Sales Data Requirements

Sales data (SALES.csv) operates with the assumption that the source system for the data is an auditing system (like Oracle Sales Audit) or non-Oracle data warehouse system. It applies minimal transformations to the inputs and assumes all the needed cleansing and preparation of transaction data has happened outside of RI. Whether you are sourcing history data from one of those systems or directly from a POS or non-Oracle auditing application, there are some business rules that should be followed.

Requirement	File Type	Explanation
Sales Units	Historical and Ongoing	The values provided for unit quantities should represent the total transaction-line values for an item, split across the gross sales units and return units. In the case of an exchange, you could have both sales and return units on the same line in RI, but most of the time only SLS or RET fields will have a value. These values will be subtracted from each other to display Net Sales Quantity metrics in RI, so they should almost always be positive.
Sales Retail Amounts	Historical and Ongoing	The retail amounts on a sale or return represent the actual selling/return value, after all discounts are subtracted from the base price of the item. In the case of an exchange, you could have both sales and return units on the same line in RI, but most of the time only SLS or RET fields will have a value. These values will be subtracted from each other to display Net Sales Amount metrics in RI, so they should almost always be positive.

Requirement	File Type	Explanation
Sales Profit Amounts	Historical and Ongoing	Profit calculations must take into consideration the cost of the item at the time it was sold, and will vary based on the retailer's costing methods. The standard approach is to use the value for Weighted Average Cost (WAC) multiplied by the units sold/returned, and subtract that total cost value from the retail amount. An item that is sold and later returned may not have the same profit amounts, if the cost has changed between the two transactions or the return was not for the full price of the item. Most POS systems do not track item costs, so providing this data requires an external process to do the calculations for you.
Sales Taxes	Historical and Ongoing	Tax amounts generally represent Value Added Tax (VAT); however this column could be used to capture other tax amounts if loading directly from the POS or external audit system.
Employee Discounts	Historical and Ongoing	These columns are specifically for employee discounts when the employee purchases a product at the POS and gets a special discount (or has the discounted amount returned later on). These values are just the discount amount, meaning the reduction in value from the selling price.
Promotional Discounts	Historical and Ongoing	These values represent the total discount taken off the initial selling price for the line-item in the transaction. These values will almost always be populated for promotional sales. However, a regular or clearance sale could have a further discount applied (like a coupon) and that should also be captured here. These values are used to populate the Sales Promotion fact table for retail type “P” transactions. So make sure that any change in price related to a promotion is included in this discount amount, so that it is copied into other tables for Promotion-level reporting.
Liabilities	Historical and Ongoing	Liabilities are sales that have not yet been charged to the customer, either due to layaway practices or customer orders that are posted as a liability transaction before they are fulfilled. Liabilities are posted with a positive value when incurred, and reversed with a negative value when they are converted into a regular sale or cancelled. Liabilities are a separate set of metrics in RI and do not interact with sales values, as it is expected that the liability will always result in a cancellation or a sale being posted at a later date.
Liability Cancels	Historical and Ongoing	Liabilities that are cancelled should first be reversed and then posted to these fields as a positive amount. A cancelled liability will have no impact on sales and has a separate set of metrics in RI. The retailer can use liability cancellation metrics to track the value of customer orders that were cancelled before being charged.

Requirement	File Type	Explanation
Retail Type	Historical and Ongoing	The retail type represents the category of sale as one of Regular, Promotion, or Clearance. We use the codes R/P/C to denote this value. The priority order when assigning a code is C > P > R, meaning that a clearance sale will always have a clearance type, even if it is also affected by a promotion. This matches the financial practices of RMFCS and Sales Audit, which treat all clearance sales as clearance activity on the stock ledger.
Transaction Reversals and Revisions	Historical and Ongoing	When using Sales Audit to audit sales, the export process will automatically handle reversing a transaction and posting revisions to a transaction. Without that, you must manually create a process to send reversals and revisions to RI matching the same data format. These two records come at the same time. A reversal is an exact opposite of the original transaction line (usually all negative values, unless the original value was negative). This will be added to RI's data and zero it out. The revision record should come next and contain the current actual values on the transaction (not the delta or difference in values). Keep in mind that, depending on the method used for calculating sales cost/profit amounts, a reversal and revision may have different values from the original profit amount. This could result in a very small residual profit from the prior revision.
Original Selling Locations	Historical and Ongoing	When including the original selling location of a return transaction, you must also make sure that is a real location included on your Organization input data. Some source systems allow the manual entry of a return's original selling location, so ensure that all such locations are included, or the records will be rejected by RI.

Inventory Data Requirements

Inventory data (INVENTORY.csv) has several special requirements that need to be followed when generating historical data and ongoing daily feeds, due to the way the data is stored within Retail Insights as well as the different use-cases in each of the Science and Planning applications. A retailer may not have the data in the required format in their source system, and adjustments would have to be made in the data extraction process to ensure these rules are followed.

Requirement	File Type	Explanation
Records may be needed before the item/location has any stock on hand	Historical and Ongoing	The inventory position contains fields for inventory movement, such as on-order, in-transit, and reserved quantities. As soon as any of those values may contain data for an item/location (and you intend to use those fields in RI), a record should be included for inventory, even if the actual stock on hand is still zero that day or week.

Requirement	File Type	Explanation
Zero balances must be sent after an item/location has started carrying a non-zero position	Historical and Ongoing	Inventory data is stored positionally, meaning we must maintain the current daily balance of stock on hand for every item/location. If no change comes into the system on a given day, we carry forward that balance. This means that you cannot send only non-zero values in the data files, as it is assumed the last known value is also the current value for the day or week. You must send a zero balance any time the inventory has moved from non-zero to zero.
Clearance indicator is used to show the end-of-period status of the item/location's inventory	Historical (Weekly)	Inventory data has a required column for a Clearance Flag (`Y`/`N`) to indicate for a given week what the status of that item/location's inventory is. The flag is intended to be the end-of-week clearance status of the item/location, so you should not send multiple records for the same item/location if the flag changed in the middle of the week. Send only one record with the correct flag for the end-of-week value. Default to `N` if you don't use it or don't know it.
Inventory records should continue to be sent based on an item/location's active ranging status	Historical (Weekly)	Once you begin sending inventory records for an item/location, it is ideal that you continue to send records every week even if the balance is zero, until that item/location is no longer active/ranged. This is important for out-of-stock detection and other analytical checks on inventory levels. A null or missing record is not automatically assumed to be zero.
Any change to values on the inventory position should send an update of the full record from the source system.	Ongoing (Daily)	If you are using other inventory values besides stock on hand (such as on-order or in-transit), you must ensure the extracts will send a complete record to the inventory interface when any of those values change. For example, a new item/location may carry an on-order balance or in-transit balance for several weeks before it has any stock on hand, so your extracts must trigger changes to those values, not just changes to stock on hand. Most other values on the interface are only used in RI reporting at this time.

For historical loads, this results in the following flow of data across all your files:

1. Generate the first month of week-ending inventory balances in INVENTORY.csv for all active item/locations in each week of data. Include zero balances if the item/loc is actively ranged to a location and is actively selling, so that out-of-stock detection is accurate. Load using the historical inventory load adhoc process.
2. Repeat the monthly file generation process, including sets of week-ending balances in chronological order. Remember that you cannot load inventory data out of order, and once a week is loaded you cannot go backwards to update past periods. All files must continue to include zero balances and other data as defined in the list of requirements.
3. Load every week of inventory snapshots through to the end of your historical period. If there will be a gap of time before starting nightly batches, plan to load an additional history file at a later date to catch up.
4. When you are ready to cutover to batches, you must also re-seed the positions of all item/locations that need to have an inventory record on Day 1 of nightly batch execution (same as for all positional facts in RI). This is needed to fill in any gaps where currently active item/locations are not present in the historical files but need to have an inventory

record added on day 1. Use the Seeding Adhoc process for Inventory to do this step, not the historical load.

Price Data Requirements

Pricing data (`PRICE.csv`) has several special requirements that need to be followed when generating historical data and ongoing daily feeds, due to the way the data is stored within Retail Insights as well as the different use-cases in the RI and Science applications. A retailer may not have the data in the required format in their source system, and adjustments would have to be made in the data extraction process to ensure these rules are followed.

Requirement	File Type	Explanation
The first price file must include records for all active item/ locations	Historical	The pricing fact is stored positionally, meaning that it first needs a starting position for an entity (for example, an initial price for an item/location) and then it may not be sent again unless there is a change to the value. The very first price file loaded into RI must contain a starting position for all active item/locations. How you determine which item/locations were active on that day in history will depend on your source system (for example, you can base it on items having stock on hand, historical ranging status, or a price history table if you have one).
The first record sent for an item/ location must come as a new price transaction type	Historical and Ongoing	Price records have a column for the price type (`PRICE_CHANGE_TRAN_TYPE`) with a fixed list of possible values. The value `0` represents a new price, which means it is the first time our system is getting a price for this item/location. All item/locations must be given a type `0` record as their first entry in the historical or ongoing data files. All the initial position records in the first file will have `type=0`. Also, all new item/locations coming in later files must first come with `type=0`.
Price records should follow a specific lifecycle using the type codes	Historical and Ongoing	The typical flow of price changes that will occur for an item/location should be as follows: • New price/cost (`PRICE_CHANGE_TRAN_TYPE=0`) • Regular cost changes (`PRICE_CHANGE_TRAN_TYPE=2`) • Regular price changes (`PRICE_CHANGE_TRAN_TYPE=4`) • Promotional/temporary markdowns (`PRICE_CHANGE_TRAN_TYPE=9`) • Clearance markdowns (`PRICE_CHANGE_TRAN_TYPE=8`) • Price reset due to new selling cycle or season change (`PRICE_CHANGE_TRAN_TYPE=0`) An item/location may have many changes with types `2/4/8/9` before eventually staying at `8` (for a final markdown) or resetting to `0` (if the item lifecycle should restart).

Requirement	File Type	Explanation
Price changes are for the end-of-day value only	Historical and Ongoing	An item price may change many times a day, but you must only send the end-of-day final position for the item/location. The file interface assumes only one record will be sent per item/location/effective date, representing the final price on that date.

For historical loads, this results in the following flow of data across all your files:

1. Generate an initial position `PRICE.csv` that has all `type=0` records for the item/locations you want to specify a starting price for. Load this as the very first file using the historical load ad hoc process.
2. Generate your first month of price change records. This will have a mixture of all the price change types. New item/location records may come in with `type=0` and records already established can get updates using any of the other type codes. Only send records when a price or cost value changes; do not send every item/location on every date. You also must not send more than one change per item/location/date.
3. Repeat the monthly file generation (or more than one month if your data volume for price changes is low) and load process until all price history has been loaded for the historical timeframe.
4. When you are ready for the cutover to batches, you must also re-seed the positions of all item/locations that need a price record on Day 1 of nightly batch execution (same as for all positional facts in RI). This is needed to fill in any gaps where currently active item/locations are not present in the historical files, but need a price record added on day 1. Use the Seeding Ad Hoc process for Pricing to do this step, not the historical load.

Other Fact File Considerations

The following section describes additional data considerations that may apply to your implementation depending on the types and volumes of data being provided to the platform. Review each topic closely, as it affects the data provided in the foundation files.

Positional Data Handling

The largest sets of fact data in the platform tend to be those that represent every possible item/location combination (such as prices or costs). To efficiently store and process these data volumes, a data warehouse technique known as compression is used to capture only the changed records on a day-to-day basis, effectively maintaining a “current position” for every set of identifiers, which is updated during each batch execution. The output of this compression process is called positional data, and the following functional areas use this method of data load and storage:

- Inventory (INV and INVU)
- Prices (PRICE)
- Costs (BCOST and NCOST)
- Purchase Orders (PO_ONORD) and Allocations On Order (PO_ONALC)

Positional data loads follow very specific rules and cannot be processed in the same manner as non-positional data such as sales transactions.

Table 7-11 Positional Data Rules

Rule	Explanation
Data Must be Sequential	Positional data must be loaded in the order of the calendar date on which it occurs and cannot be loaded out-of-order. For example, when loading history data for inventory, you must provide each week of inventory one after the other, starting from Week 1, 2, 3, and so on.
Data Cannot be Back Posted	Positional data cannot be posted to any date prior to the current load date or business date of the system. If your current load date is Week 52 2021, you cannot post records back to Week 50: those past positions are unable to be changed. Any corrections that need to be loaded must be effective from the current date forward.
Data Must be Seeded	Because positional data must maintain the current position of all data elements in the fact (even those that are inactive or not changing) it is required to initialize or "seed" positional facts with a starting value for every possible combination of identifiers. This happens at two times: 1. The first date in your history files must be full snapshots of all item/locations that need a value, including zero balances for things like inventory. 2. Special seed programs are provided to load initial full snapshots of data after history is finished, to prepare you for nightly batch runs. After seeding, you are allowed to provide incremental datasets (posting only the positions that change, not the full daily or weekly snapshot). Incremental loads are one of the main benefits of using positional data, as they greatly reduce your nightly batch runtime.

Throughout the initial data load process, there will be additional steps called out any time a positional load must be performed, to ensure you accurately capture both historical and initial seed data before starting nightly batch runs.

System Parameters File

The dimension file for `RA_SRC_CURR_PARAM_G.dat` is not used as part of your history load process directly, but instead provides an important piece of information to the platform for operational activities. This file must contain the current business date associated with the files in the ZIP package. The file should be included with the nightly ZIP upload that contains your foundation data, such as `RI_RMS_DATA.zip` or `RAP_DATA.zip`.

The file has only two generic columns, `PARAM_NAME` and `PARAM_VALUE`. When data is sourced from RMFCS, it will be automatically generated and sent to RI in the nightly batch. If your data does not come from RMFCS, then you need to include the file manually. Currently, only two rows will be used, but future releases may look for additional parameters in the file.

The file should contain the following rows:

```
VDATE|20220101
PRIME_CURRENCY_CODE|USD
```

The parameter value with `VDATE` is the current business date that all your other files were generated for in YYYYMMDD format. The date should match the values on your

fact data, such as the DAY_DT columns in sales, inventory, and so on. This format is not configurable and should be provided as shown. The parameter value with PRIME_CURRENCY_CODE is used by the system to set default currencies on fact files when you do not provide them yourself or if there are null currency codes on a row.

Loading Data on Multiple Threads

Large data files, such as sales and inventory, can support multiple threads for parallel execution, which can greatly reduce the load times for history files. Some tables have an explicit column called ETL_THREAD_VAL to define your threads for incoming data. Parameters are available in C_ODI_PARAM_VW (in the Control Center) to configure each load program's maximum number of threads, using the parameter name LOC_NUM_OF_THREAD. For larger retailers, it is common to define 5 to 8 threads for incoming data feeds. You may choose how you split data across threads when generating the files, but a common strategy is to assign stores to threads, such that all the data for a given location is processed on the same thread.

For many interfaces, the ETL_THREAD_VAL column has been removed and, in those cases, the system will automatically calculate the threads based on the LOC_NUM_OF_THREAD parameter values. If ETL_THREAD_VAL is not provided on a file that has the column, the same automation will be applied to determine multi-threading behavior. The LOC_NUM_OF_THREAD values must be updated appropriately by the implementer in C_ODI_PARAM to leverage multi-threading capabilities.

A

Legacy Foundation File Reference

The following table provides a cross-reference for legacy application input files and the Retail Analytics Platform files that replace them. This list covers foundation data flows which span multiple applications, such as MFP and RI. Other foundation files exist which do not replace multiple application files; those are specified in the Interfaces Guide in My Oracle Support.

File Group	File Type	Legacy Planning Files	Legacy RI/RSP Files	RAP Files
Product	Dimension	prod.csv.dat	W_PRODUCT_DS.dat W_PRODUCT_DS_TL.dat W_PROD_CAT_DHS.dat W_DOMAIN_MEMBER_DS_TL.dat W_RTL_PRODUCT_BRAND_DS.dat W_RTL_PRODUCT_BRAND_DS_TL.dat W_RTL_IT_SUPPLIER_DS.dat W_PARTY_ORG_DS.dat W_RTL_PRODUCT_IMAGE_DS.dat W_PRODUCT_ATTR_DS.dat W_RTL_ITEM_GRP1_DS.dat	PRODUCT.csv
Organization	Dimension	loc.csv.dat stor_metrics.csv.ovr	W_INT_ORG_DS.dat W_INT_ORG_DS_TL.dat W_INT_ORG_DHS.dat W_DOMAIN_MEMBER_DS_TL.dat (for RTL_ORG) W_RTL_CHANNEL_DS.dat W_INT_ORG_ATTR_DS.dat	ORGANIZATION.csv
Calendar	Dimension	clnd.csv.dat	W_MCAL_PERIOD_DS.dat	CALENDAR.csv
Exchange Rates	Dimension	curh.csv.dat curr.csv.ovr	W_EXCH_RATE_GS.dat	EXCH_RATE.csv
Attributes	Dimension	patr.csv.dat patv.csv.ovr	W_RTL_PRODUCT_ATTR_DS.dat W_RTL_PRODUCT_ATTR_DS_TL.dat W_DOMAIN_MEMBER_DS_TL.dat (for Diffs) W_RTL_PRODUCT_COLOR_DS.dat	ATTR.csv
Diff Groups	Dimension	sizh.hdr.csv.dat	W_RTL_DIFF_GRP1_DS.dat W_RTL_DIFF_GRP1_DS_TL.dat	DIFF_GROUP.csv

File Group	File Type	Legacy Planning Files	Legacy RI/RSP Files	RAP Files
Product Attribute Assignments	Dimension	prdatt.csv.ovr	W_RTL_ITEM_GRP1_DS.dat	PROD_ATTR.csv
Sales	Fact	rsal.csv.ovr psal.csv.ovr csal.csv.ovr nsls.csv.ovr rtn.csv.ovr	W_RTL_SLS_TRX_IT_LC_DY_FS.dat W_RTL_SLSPK_IT_LC_DY_FS.dat	SALES.csv SALES_PACK.csv
Inventory	Fact	eop.csv.ovr eopx.csv.ovr wsal.csv.ovr	W_RTL_INV_IT_LC_DY_FS.dat	INVENTORY.csv
Markdown	Fact	mkd.csv.ovr	W_RTL_MKDN_IT_LC_DY_FS.dat	MARKDOWN.csv
On Order	Fact	oo.csv.ovr	W_RTL_PO_DETAILS_DS.dat W_RTL_PO_ONDORD_IT_LC_DY_FS.dat	ORDER_HEAD.csv ORDER_DETAIL.csv
PO Receipts	Fact	rcpt.csv.ovr	W_RTL_INVRC_IT_LC_DY_FS.dat	RECEIPT.csv
Transfers	Fact	tranx.csv.ovr	W_RTL_INVTSF_IT_LC_DY_FS.dat	TRANSFER.csv
Adjustments	Fact	tran.csv.ovr	W_RTL_INVADJ_IT_LC_DY_FS.dat W_REASON_DS.dat	ADJUSTMENT.csv REASON.csv
RTVs	Fact	tran.csv.ovr	W_RTL_INVRTV_IT_LC_DY_FS.dat	RTV.csv
Costs	Fact	slsprc.csv.ovr	W_RTL_BCOST_IT_LC_DY_FS.dat W_RTL_NCOST_IT_LC_DY_FS.dat	COST.csv
Prices	Fact	slsprc.csv.ovr	W_RTL_PRICE_IT_LC_DY_FS.dat	PRICE.csv
W/F Sales and Fees	Fact	tran.csv.ovr	W_RTL_SLSWF_IT_LC_DY_FS.dat	SALES_WF.csv
Vendor Funds (TC 6/7)	Fact	tran.csv.ovr	W_RTL_DEALINC_IT_LC_DY_FS.dat	DEAL_INCOME.csv
Reclass In/Out (TC 34/36)	Fact	tran.csv.ovr	W_RTL_INVRECLASS_IT_LC_DY_FS.dat	INV_RECLASS.csv
Intercompany Margin (TC 39)	Fact	tran.csv.ovr	W_RTL_ICM_IT_LC_DY_FS.dat	IC_MARGIN.csv

B

Context File Table Reference

The following table maps CSV data files to internal tables for the purpose of creating Context Files. The first parameter on the Context file is a `TABLE` property containing the table name into which the CSV data will be loaded. For legacy context file usage, the name of the context file itself should match the internal table name.

File Type	Filenames	Internal Tables
Product	PRODUCT.csv	W_PRODUCT_DTS
Organization	ORGANIZATION.csv	W_INT_ORG_DTS
Calendar	CALENDAR.csv	W_MCAL_PERIODS_DTS
Exchange Rates	EXCH_RATE.csv	W_EXCH_RATE_DTS
Attributes	ATTR.csv	W_ATTR_DTS
Diff Groups	DIFF_GROUP.csv	W_DIFF_GROUP_DTS
Product Attribute Assignments	PROD_ATTR.csv	W_PRODUCT_ATTR_DTS
Employee	EMPLOYEE.csv	W_EMPLOYEE_DTS
Application Codes	CODES.csv	W_RTL_CODE_DTS
Pack Items	PROD_PACK.csv	W_RTL_ITEM_GRP2_DTS
Promotions	PROMOTION.csv	W_RTL_PROMO_EXT_DTS
Supplier	SUPPLIER.csv	W_SUPPLIER_DTS
Item Loc Attributes	PROD_LOC_ATTR.csv	W_PROD_LOC_ATTR_DTS
Replenishment Attributes	PROD_LOC_REPL.csv	W_INVENTORY_PRODUCT_ATTR_DTS
Season Phase	SEASON.csv	W_RTL_SEASON_PHASE_DTS
Season Phase Item Mapping	PROD_SEASON.csv	W_RTL_SEASON_PHASE_IT_DTS
Comp Stores	STORE_COMP.csv	W_RTL_LOC_COMP_MTX_DTS
Item Deletes	PROD_DELETE.csv	W_RTL_ITEM_DEL_TMPS
Item Loc Deletes	PROD_LOC_DELETE.csv	W_RTL_IT_LC_DEL_TMPS
Item Reclass	PROD_RECLASS.csv	W_PROD_RECLASS_TMPS
Sales	SALES.csv	W_RTL_SLS_TRX_IT_LC_DY_FTS
Sales Pack	SALES_PACK.csv	W_RTL_SLSPK_IT_LC_DY_FTS
Inventory	INVENTORY.csv	W_RTL_INV_IT_LC_DY_FTS
Markdown	MARKDOWN.csv	W_MARKDOWN_FTS
On Order	ORDER_HEAD.csv	W_ORDER_HEAD_FTS
On Order Detail	ORDER_DETAIL.csv	W_ORDER_DETAIL_FTS
PO Receipts	RECEIPT.csv	W_RECEIPT_FTS
Transfers	TRANSFER.csv	W_RTL_INVTSF_IT_LC_DY_FTS
Adjustments	ADJUSTMENT.csv	W_ADJUSTMENT_FTS

File Type	Filenames	Internal Tables
RTVs	RTV.csv	W_RTL_INVRTV_IT_LC_DY_FTS
Costs	COST.csv	W_COST_FTS
Prices	PRICE.csv	W_RTL_PRICE_IT_LC_DY_FTS
W/F Sales and Fees	SALES_WF.csv	W_RTL_SLSWF_IT_LC_DY_FTS
Vendor Funds	DEAL_INCOME.csv	W_RTL_DEALINC_IT_LC_DY_FTS
Reclass In/Out	INV_RECLASS.csv	W_RTL_INVRECLASS_IT_LC_DY_FTS
Intercompany Margin	IC_MARGIN.csv	W_RTL_ICM_IT_LC_DY_FTS

C

Sample Public File Transfer Script for Planning Apps

This example provides an example of how file transfers can be implemented through a shell script. It requires: bash, curl and jq.

```
#!/bin/bash

# Sample Public FTS script

### EDIT HERE to reflect your environment

BASE_URL="https://__YOUR_TENANT_BASE_URL__"
TENANT="__YOUR-TENANT_ID__"
IDCS_URL="https://_YOUR__IDCS__URL__/oauth2/v1/token"
IDCS_CLIENTID="__YOUR_CLIENT_APPID__"
IDCS_CLIENTSECRET="__YOUR_CLIENT_SECRET___"
IDCS_SCOPE=”rgbu:rpas:psraf-__YOUR_SCOPE__”

### FINISHED

clientToken() {
    curl  -sX POST "${IDCS_URL}" \
          --header "Authorization: Basic ${IDCS_AUTH}" \
          --header "Content-Type: application/x-www-form-urlencoded" \
          --data-urlencode "grant_type=client_credentials" \
          --data-urlencode "scope=${IDCS_SCOPE}"  | jq -r .access_token
}

ping() {
    echo "Pinging"
    curl -sfX GET "${BASE_URL}/${TENANT}/RetailAppsReSTServices/services/
private/FTSWrapper/ping" \
         --header 'Accept: application/json' \
         --header 'Accept-Language: en' \
         --header "Authorization: Bearer ${CLIENT_TOKEN}" | jq
}


listPrefixes() {
    echo "Listing storage prefixes"
    curl -sfX GET "${BASE_URL}/${TENANT}/RetailAppsReSTServices/services/
private/FTSWrapper/listprefixes" \
         --header 'Accept: application/json' \
         --header 'Accept-Language: en' \
         --header "Authorization: Bearer ${CLIENT_TOKEN}" | jq
}
```

```
listFiles() {
    echo "Listing files for ${1}"
    curl -sfX GET "${BASE_URL}/${TENANT}/RetailAppsReSTServices/
services/private/FTSWrapper/listfiles?prefix=${1}" \
         --header 'Accept: application/json' \
         --header 'Accept-Language: en' \
         --header "Authorization: Bearer ${CLIENT_TOKEN}" | jq
}

deleteFiles() {
    echo "Deleting files"
    json=$(fileCollection $@)
    curl --show-error -sfX DELETE "${BASE_URL}/${TENANT}/
RetailAppsReSTServices/services/private/FTSWrapper/delete" \
         --header 'content-type: application/json' \
         --header 'Accept: application/json' \
         --header 'Accept-Language: en' \
         --header "Authorization: Bearer ${CLIENT_TOKEN}" \
         -d "${json}" | jq
}

fileMover() {
    movement="${1}"
    shift

    json=$(fileCollection $@)
    requestPAR "${movement}" "${json}"
}

fileCollection() {
    local json="{  \"listOfFiles\": [ __FILES__ ] }"

    sp="${1}"
    shift

    while (( "${#}" )); do
        list="${list} { \"storagePrefix\": \"${sp}\", \"fileName\":
\"${1}\" }"
        if [[ ${#} -gt "1" ]]; then
            list="${list},"
        fi
        shift
    done

    echo "${json/__FILES__/${list}}"
}

requestPAR() {
    use="${1}"
    echo "Requesting PARs for ${use}"
    pars="$(curl --show-error -sfX POST "${BASE_URL}/${TENANT}/
RetailAppsReSTServices/services/private/FTSWrapper/${use}" \
         --header 'content-type: application/json' \
         --header 'Accept: application/json' \
```

```
         --header 'Accept-Language: en' \
         --header "Authorization: Bearer ${CLIENT_TOKEN}" \
         -d "${2}")"

    if [[ "$?" -ne "0" ]]; then
        echo "Error retreiving PAR, check files specified."
        echo "${pars}"
        exit 1
    fi

    list=$(jq -r ".parList[]|[.name, .accessUri]|@csv" <<< "${pars}")

    while IFS='' read -r line; do
        fn=$(echo ${line}|cut -d\, -f1|tr -d \")
        par=$(echo ${line}|cut -d\, -f2|tr -d \")

        if [[ ${use} == "upload" ]]; then
            echo "Uploading ${fn} to ${par}"
            curl -o log -X PUT --data-binary "@${fn}" "${par}"
        else
            echo "Downloading ${fn} from ${par}"
            curl -o ${fn} -X GET "${par}"
        fi
    done <<< "${list}"
}


#Entry point
IDCS_AUTH=$(echo -n ${IDCS_CLIENTID}:${IDCS_CLIENTSECRET} | base64 -w0)
CLIENT_TOKEN=$(clientToken)

case "${1}" in
    ping)
        ping
        ;;
    listprefixes)
        shift
        listPrefixes
        ;;
    listfiles)
        shift
        listFiles ${@}
        ;;
    deletefiles)
        shift
        deleteFiles ${@}
        ;;
    uploadfiles)
        shift
        fileMover upload ${@}
        ;;
    downloadfiles)
        shift
        fileMover download ${@}
        ;;
```

```
    *)
        echo "Usage: $0"
        echo " ping                                          : test
service functionality"
        echo " listprefixes                                  : list
registered prefixes"
        echo " listfiles [prefix]                            : list files
within a prefix"
        echo " deletefiles [prefix] [file1] [file2] ...   : delete
files with this prefix"
        echo " uploadfiles [prefix] [file1] [file2] ...   : upload
files with this prefix"
        echo " downloadfiles [prefix] [file1] [file2] ... : download
files with this prefix"
        echo
        exit 0
        ;;
esac
```

D

Sample Public File Transfer Script for RI and RSP

```
#!/bin/bash

# Sample Public FTS script

### EDIT HERE to reflect your environment

BASE_URL="https://__YOUR_TENANT_BASE_URL__"
TENANT="__YOUR-TENANT_ID__"
IDCS_URL="https://_YOUR__IDCS__URL__/oauth2/v1/token"
IDCS_CLIENTID="__YOUR_CLIENT_APPID__"
IDCS_CLIENTSECRET="__YOUR_CLIENT_SECRET___"
IDCS_SCOPE=”rgbu:rsp:psraf-__YOUR_SCOPE__”

### FINISHED

clientToken() {
    curl  -sX POST "${IDCS_URL}" \
          --header "Authorization: Basic ${IDCS_AUTH}" \
          --header "Content-Type: application/x-www-form-urlencoded" \
          --data-urlencode "grant_type=client_credentials" \
          --data-urlencode "scope=${IDCS_SCOPE}"  | jq -r .access_token
}

ping() {
    echo "Pinging"
    curl -sfX GET "${BASE_URL}/${TENANT}/RIRetailAppsReSTServices/services/
private/FTSWrapper/ping" \
         --header 'Accept: application/json' \
         --header 'Accept-Language: en' \
         --header "Authorization: Bearer ${CLIENT_TOKEN}" | jq
}


listPrefixes() {
    echo "Listing storage prefixes"
    curl -sfX GET "${BASE_URL}/${TENANT}/RIRetailAppsReSTServices/services/
private/FTSWrapper/listprefixes" \
         --header 'Accept: application/json' \
         --header 'Accept-Language: en' \
         --header "Authorization: Bearer ${CLIENT_TOKEN}" | jq
}


listFiles() {
    echo "Listing files for ${1}"
```

```
    curl -sfX GET "${BASE_URL}/${TENANT}/RIRetailAppsReSTServices/
services/private/FTSWrapper/listfiles?prefix=${1}" \
         --header 'Accept: application/json' \
         --header 'Accept-Language: en' \
         --header "Authorization: Bearer ${CLIENT_TOKEN}" | jq
}

deleteFiles() {
    echo "Deleting files"
    json=$(fileCollection $@)
    curl --show-error -sfX DELETE "${BASE_URL}/${TENANT}/
RIRetailAppsReSTServices/services/private/FTSWrapper/delete" \
         --header 'content-type: application/json' \
         --header 'Accept: application/json' \
         --header 'Accept-Language: en' \
         --header "Authorization: Bearer ${CLIENT_TOKEN}" \
         -d "${json}" | jq
}

fileMover() {
    movement="${1}"
    shift

    json=$(fileCollection $@)
    requestPAR "${movement}" "${json}"
}

fileCollection() {
    local json="{  \"listOfFiles\": [ __FILES__ ] }"

    sp="${1}"
    shift

    while (( "${#}" )); do
        list="${list} { \"storagePrefix\": \"${sp}\", \"fileName\":
\"${1}\" }"
        if [[ ${#} -gt "1" ]]; then
            list="${list},"
        fi
        shift
    done

    echo "${json/__FILES__/${list}}"
}

requestPAR() {
    use="${1}"
    echo "Requesting PARs for ${use}"
    pars="$(curl --show-error -sfX POST "${BASE_URL}/${TENANT}/
RIRetailAppsReSTServices/services/private/FTSWrapper/${use}" \
         --header 'content-type: application/json' \
         --header 'Accept: application/json' \
         --header 'Accept-Language: en' \
         --header "Authorization: Bearer ${CLIENT_TOKEN}" \
         -d "${2}")"
```

```
    if [[ "$?" -ne "0" ]]; then
        echo "Error retreiving PAR, check files specified."
        echo "${pars}"
        exit 1
    fi

    list=$(jq -r ".parList[]|[.name, .accessUri]|@csv" <<< "${pars}")

    while IFS='' read -r line; do
        fn=$(echo ${line}|cut -d\, -f1|tr -d \")
        par=$(echo ${line}|cut -d\, -f2|tr -d \")

        if [[ ${use} == "upload" ]]; then
            echo "Uploading ${fn} to ${par}"
            curl -o log -X PUT --data-binary "@${fn}" "${par}"
        else
            echo "Downloading ${fn} from ${par}"
            curl -o ${fn} -X GET "${par}"
        fi
    done <<< "${list}"
}


#Entry point
IDCS_AUTH=$(echo -n ${IDCS_CLIENTID}:${IDCS_CLIENTSECRET} | base64 -w0)
CLIENT_TOKEN=$(clientToken)

case "${1}" in
    ping)
        ping
        ;;
    listprefixes)
        shift
        listPrefixes
        ;;
    listfiles)
        shift
        listFiles ${@}
        ;;
    deletefiles)
        shift
        deleteFiles ${@}
        ;;
    uploadfiles)
        shift
        fileMover upload ${@}
        ;;
    downloadfiles)
        shift
        fileMover download ${@}
        ;;
    *)
        echo "Usage: $0"
        echo " ping                                             : test service
```

```
functionality"
        echo " listprefixes                                 : list
registered prefixes"
        echo " listfiles [prefix]                           : list files
within a prefix"
        echo " deletefiles [prefix] [file1] [file2] ...     : delete
files with this prefix"
        echo " uploadfiles [prefix] [file1] [file2] ...     : upload
files with this prefix"
        echo " downloadfiles [prefix] [file1] [file2] ... : download
files with this prefix"
        echo
        exit 0
        ;;
esac
```

E

Sample Validation SQLs

This set of sample SQL commands provides scripts to run using APEX which can help validate your initial dimension and fact loads, especially if it is the first time loading the data and quality is unknown. Do not load data into the platform without performing some level of validation on it first, as this will greatly reduce the time spent reworking and reloading data.

```
------------------------------------------------
-- Checks for CALENDAR.csv file load
------------------------------------------------
-- Verify initial calendar data before staging it further, row counts should
match data file
SELECT * FROM W_MCAL_PERIOD_DTS

-- Check total counts, all counts should be same. This can indirectly check
for nulls in required columns.
SELECT
count(*),count(MCAL_CAL_ID),count(MCAL_PERIOD_TYPE),count(MCAL_PERIOD_NAME),
count(MCAL_PERIOD),count(MCAL_PERIOD_ST_DT),count(MCAL_PERIOD_END_DT),count(M
CAL_QTR),
count(MCAL_YEAR),count(MCAL_QTR_START_DT),count(MCAL_QTR_END_DT),count(MCAL_Y
EAR_START_DT),
count(MCAL_YEAR_END_DT) FROM W_MCAL_PERIOD_DTS

-- This should not return any rows
SELECT * FROM W_MCAL_PERIOD_DTS WHERE MCAL_CAL_ID IS NULL or MCAL_CAL_ID !=
'Retail Calendar~41'

-- Checking duplicate rows, if any. This should not return any rows.
SELECT MCAL_PERIOD_NAME,count(*) FROM W_MCAL_PERIOD_DTS GROUP BY
MCAL_PERIOD_NAME having count(MCAL_PERIOD_NAME) > 1


-- Check number of periods per year. Should always be 12.
SELECT MCAL_YEAR,count(MCAL_PERIOD_NAME) FROM W_MCAL_PERIOD_DTS GROUP BY
MCAL_YEAR ORDER BY MCAL_YEAR
SELECT MCAL_YEAR,count(MCAL_PERIOD) FROM W_MCAL_PERIOD_DTS GROUP BY
MCAL_YEAR ORDER BY MCAL_YEAR

-- After Load procedures completed, check following tables
select count(*) from W_MCAL_PERIOD_D
select count(*) from W_MCAL_DAY_D
select count(*) from W_MCAL_WEEK_D

------------------------------------------------
-- Checks for PRODUCT.csv file load
------------------------------------------------
```

```
-- Verify initial product data before staging it further, row counts
should match data file
select * from W_PRODUCT_DTS

-- Check total count, all counts should be same.  This can indirectly
check for nulls in required columns.
SELECT
count(*),count(item),count(distinct(item)),count(item_level),count(tran
_level),
count(LVL4_PRODCAT_ID),count(LVL4_PRODCAT_UID),count(LVL5_PRODCAT_ID),c
ount(LVL5_PRODCAT_UID),
count(LVL6_PRODCAT_ID),count(LVL7_PRODCAT_ID),count(LVL8_PRODCAT_ID),co
unt(TOP_PRODCAT_ID),
count(ITEM_DESC),count(LVL4_PRODCAT_DESC),count(LVL5_PRODCAT_DESC),coun
t(LVL6_PRODCAT_DESC),
count(LVL7_PRODCAT_DESC),count(LVL8_PRODCAT_DESC),count(TOP_PRODCAT_DES
C) FROM W_PRODUCT_DTS

-- Check individual counts to make sure it aligns with your source data
SELECT
count(*),count(ITEM_PARENT),count(distinct(ITEM_PARENT)),count(ITEM_GRA
NDPARENT),count(distinct(ITEM_GRANDPARENT)) FROM W_PRODUCT_DTS WHERE
ITEM_LEVEL = 1
SELECT
count(*),count(ITEM_PARENT),count(distinct(ITEM_PARENT)),count(ITEM_GRA
NDPARENT),count(distinct(ITEM_GRANDPARENT)) FROM W_PRODUCT_DTS WHERE
ITEM_LEVEL = 2
SELECT
count(*),count(ITEM_PARENT),count(distinct(ITEM_PARENT)),count(ITEM_GRA
NDPARENT),count(distinct(ITEM_GRANDPARENT)) FROM W_PRODUCT_DTS WHERE
ITEM_LEVEL = 3

-- Checking duplicate rows, if any. This should not return any rows.
SELECT item,count(1) FROM W_PRODUCT_DTS GROUP BY item having count(1)
> 1

-- Check item_level, should not have NULL, should have values only 1,2
or 3. Make sure Count makes sense
SELECT item_level, count(*) FROM W_PRODUCT_DTS GROUP BY item_level
ORDER BY 1

-- Check tran_level, should not have NULL, should have only one value
for our purpose. Make sure Count makes sense
SELECT tran_level, count(*) FROM W_PRODUCT_DTS GROUP BY tran_level
ORDER BY 1

-- After DTS to DS job executed, check following tables for data
SELECT count(*) FROM W_PRODUCT_DS
SELECT count(*) FROM W_PRODUCT_DS_TL
SELECT count(*) FROM W_PROD_CAT_DHS
SELECT count(*) FROM W_DOMAIN_MEMBER_DS_TL

-- Expect records for "MCAT" which is the product hierarchy labels code
SELECT DOMAIN_CODE, DOMAIN_TYPE_CODE,LANGUAGE_CODE,
SRC_LANGUAGE_CODE,count(1)
```

```
FROM W_DOMAIN_MEMBER_DS_TL GROUP BY DOMAIN_CODE,
DOMAIN_TYPE_CODE,LANGUAGE_CODE, SRC_LANGUAGE_CODE

-- After Load procedures completed, check following tables
select count(*) from w_prod_cat_dh
select count(*) from w_product_d
select count(*) from w_product_d_tl

-- check for MCAT records for hierachy labels, should align with hierachy
level counts
select domain_code,count(*) from W_DOMAIN_MEMBER_LKP_TL group by domain_code


-----------------------------------------------
-- Checks for ORGANIZATION.csv file load
-----------------------------------------------
-- Verify initial location data before staging it further, row counts should
match data file
SELECT * FROM W_INT_ORG_DTS

-- Check total count, all counts should be same.  This can indirectly check
for nulls in required columns.
SELECT
count(*),count(ORG_NUM),count(distinct(ORG_NUM)),count(ORG_TYPE_CODE),count(C
URR_CODE),
count(ORG_HIER10_NUM),count(ORG_HIER11_NUM),count(ORG_HIER12_NUM),count(ORG_H
IER13_NUM),
count(ORG_TOP_NUM),count(ORG_DESC),count(ORG_HIER10_DESC),count(ORG_HIER11_DE
SC),
count(ORG_HIER12_DESC),count(ORG_HIER13_DESC),count(ORG_TOP_DESC) FROM
W_INT_ORG_DTS

-- Checking duplicate rows, if any. This should not return any rows.
SELECT ORG_NUM,count(1) FROM W_INT_ORG_DTS GROUP BY ORG_NUM having count(1)
> 1

-- Check ORG_TYPE_CODE, CURR_CODE should not have nulls
-- ORG_TYPE_CODE should be either "S" for Store or "W" for Warehouse
SELECT ORG_TYPE_CODE, CURR_CODE, count(*) FROM W_INT_ORG_DTS GROUP BY
ORG_TYPE_CODE, CURR_CODE ORDER BY 2,1

-- After DTS to DS job executed, check following tables for expected data
SELECT count(*) FROM W_INT_ORG_DS
SELECT count(*) FROM W_INT_ORG_DS_TL
SELECT count(*) FROM W_INT_ORG_DHS
SELECT count(*) FROM W_DOMAIN_MEMBER_DS_TL
-- Expect records for "RTL_ORG" which is the location hierarchy labels code
SELECT DOMAIN_CODE, DOMAIN_TYPE_CODE,LANGUAGE_CODE,
SRC_LANGUAGE_CODE,count(1)
FROM W_DOMAIN_MEMBER_DS_TL GROUP BY DOMAIN_CODE,
DOMAIN_TYPE_CODE,LANGUAGE_CODE, SRC_LANGUAGE_CODE

-- Org hierarchy level validation on DHS table before loading it to DH
-- On first loading a new hierarchy, this must return with zero issues in
the level structure.  DO NOT proceed if any issues show up.
```

```
SELECT '1',
                  'LOCATION org_hier9_num' LEVEL_DESC,
                  location_a.org_hier9_num C_LEVEL,
                  location_a.org_hier10_num P1_LEVEL,
                  location_a.org_hier11_num P2_LEVEL,
                  location_a.org_hier12_num P3_LEVEL,
                  location_a.org_hier13_num P4_LEVEL,
                  location_a.org_top_num P5_LEVEL
             FROM w_int_org_dhs location_a,  w_int_org_dhs location_b
            WHERE location_a.level_name = 'LOCATION'
              AND location_b.level_name = location_a.level_name
              AND location_a.org_hier9_num = location_b.org_hier9_num
              AND (location_a.org_hier10_num <>
location_b.org_hier10_num
                  or location_a.org_hier11_num <>
location_b.org_hier11_num
                  or location_a.org_hier12_num <>
location_b.org_hier12_num
                  or location_a.org_hier13_num <>
location_b.org_hier13_num
                  or location_a.org_top_num <> location_b.org_top_num)
        UNION ALL
           SELECT '1',
                  'DISTRICT org_hier10_num' LEVEL_DESC,
                  location_a.org_hier10_num C_LEVEL,
                  null P1_LEVEL,
                  location_a.org_hier11_num P2_LEVEL,
                  location_a.org_hier12_num P3_LEVEL,
                  location_a.org_hier13_num P4_LEVEL,
                  location_a.org_top_num P5_LEVEL
             FROM w_int_org_dhs location_a,  w_int_org_dhs location_b
            where location_a.level_name = 'DISTRICT'
              and location_b.level_name = location_a.level_name
              and location_a.org_hier10_num = location_b.org_hier10_num
              and (location_a.org_hier11_num <>
location_b.org_hier11_num
                  or location_a.org_hier12_num <>
location_b.org_hier12_num
                  or location_a.org_hier13_num <>
location_b.org_hier13_num
                  or location_a.org_top_num <> location_b.org_top_num)
        UNION ALL
           SELECT '1',
                  'REGION org_hier11_num' LEVEL_DESC,
                    location_a.org_hier11_num C_LEVEL,
                  NULL P1_LEVEL,
                  NULL P2_LEVEL,
                  location_a.org_hier12_num P3_LEVEL,
                  location_a.org_hier13_num P4_LEVEL,
                  location_a.org_top_num P5_LEVEL
             FROM w_int_org_dhs location_a,  w_int_org_dhs location_b
            where location_a.level_name = 'REGION'
              and location_b.level_name = location_a.level_name
              and location_a.org_hier11_num = location_b.org_hier11_num
              and (location_a.org_hier12_num <>
```

```
location_b.org_hier12_num
                  or location_a.org_hier13_num <> location_b.org_hier13_num
                  or location_a.org_top_num <> location_b.org_top_num)
        UNION ALL
           SELECT '1',
                  'AREA org_hier12_num' LEVEL_DESC,
                  location_a.org_hier12_num C_LEVEL,
                  NULL P1_LEVEL,
                  NULL P2_LEVEL,
                  NULL P3_LEVEL,
                  location_a.org_hier13_num P4_LEVEL,
                  location_a.org_top_num P5_LEVEL
             FROM w_int_org_dhs location_a,  w_int_org_dhs location_b
            where location_a.level_name = 'AREA'
              and location_b.level_name = location_a.level_name
              and location_a.org_hier12_num = location_b.org_hier12_num
              and (location_a.org_hier13_num <> location_b.org_hier13_num
                   or location_a.org_top_num <> location_b.org_top_num)
        UNION ALL
           SELECT '1',
                  'CHAIN org_hier13_num' LEVEL_DESC,
                  location_a.org_hier13_num C_LEVEL,
                  NULL P1_LEVEL,
                  NULL P2_LEVEL,
                  NULL P3_LEVEL,
                  NULL P4_LEVEL,
                  location_a.org_top_num P5_LEVEL
             FROM w_int_org_dhs location_a,  w_int_org_dhs location_b
            where location_a.level_name = 'CHAIN'
              and location_b.level_name = location_a.level_name
              and location_a.org_hier13_num = location_b.org_hier13_num
              and location_a.org_top_num <> location_b.org_top_num;

-- After Load procedures completed, check following tables for final
dimension data
select count(*) from w_int_org_dh
select count(*) from w_int_org_d
select count(*) from w_int_org_d_tl

-- check for RTL_ORG records for the descriptions of hierarchy levels
select domain_code,count(*) from W_DOMAIN_MEMBER_LKP_TL group by domain_code


------------------------------------------------
-- Checks on EXCH_RATE.csv file load
------------------------------------------------
select * from w_exch_rate_dts

select * from w_exch_rate_gs

select * from w_exch_rate_g


------------------------------------------------
-- Checks on ATTR.csv and PROD_ATTR.csv file load
```

```
-----------------------------------------------
select * from w_attr_dts
select * from w_product_attr_dts
select * from w_rtl_item_grp1_ds
select * from w_rtl_item_grp1_d

-----------------------------------------------
-- Check on W_DOMAIN_MEMBER_LKP_TL issues while loading dimensions
-----------------------------------------------
--- DOMAIN MEMBER DUPLICATE RECORD ERROR ---
SELECT DOMAIN_CODE,DOMAIN_TYPE_CODE,DOMAIN_MEMBER_CODE,count(1) FROM
W_DOMAIN_MEMBER_DS_TL GROUP BY
DOMAIN_CODE,DOMAIN_TYPE_CODE,DOMAIN_MEMBER_CODE having count(1) > 1


SELECT DOMAIN_TYPE_CODE, DOMAIN_MEMBER_CODE, DOMAIN_MEMBER_NAME FROM
W_DOMAIN_MEMBER_DS_TL WHERE
(DOMAIN_CODE,DOMAIN_TYPE_CODE,DOMAIN_MEMBER_CODE) IN
(SELECT DOMAIN_CODE,DOMAIN_TYPE_CODE,DOMAIN_MEMBER_CODE FROM
W_DOMAIN_MEMBER_DS_TL GROUP BY
DOMAIN_CODE,DOMAIN_TYPE_CODE,DOMAIN_MEMBER_CODE having count(1) > 1)
ORDER BY 1,2,3


-----------------------------------------------
-- Checks on SALES.csv file
-----------------------------------------------
-- Verify initial sales data before staging it further, check all
columns are populated with expected values (i.e. CTX was properly
formed)
select * from W_RTL_SLS_TRX_IT_LC_DY_FTS

-- Should match the record count from last loaded SALES.csv file
select count(*) from W_RTL_SLS_TRX_IT_LC_DY_FTS

-- Should match W_RTL_SLS_TRX_IT_LC_DY_FTS count
select count(*) from W_RTL_SLS_TRX_IT_LC_DY_FS

-- Should match or be close W_RTL_SLS_TRX_IT_LC_DY_FS count
select count(*) from W_RTL_SLS_TRX_IT_LC_DY_F

-- Look for sls_trx_id in stage but not final sales table
select * from
(
select fs.fs_trx_id, tg_trx_id
from
(select /*+ parallel */ sls_trx_id fs_trx_id from
w_rtl_sls_trx_it_lc_dy_fs) fs left outer join
(select sls_trx_id tg_trx_id from w_rtl_sls_trx_it_lc_dy_f union all
select sls_trx_id from e$_w_rtl_sls_trx_it_lc_dy_tmp) tg
on (fs.fs_trx_id = tg.tg_trx_id)
) where tg_trx_id is null;

-- Look for sls_trx_id/prod_it_wid in stage but not final sales table
select * from
```

```
(
select fs.prod_it_wid, fs.fs_trx_id,  prod_wid, tg_trx_id
from
(select /*+ parallel */ prod_it_wid, sls_trx_id fs_trx_id from
w_rtl_sls_trx_it_lc_dy_tmp) fs left outer join
(select prod_wid, sls_trx_id tg_trx_id from w_rtl_sls_trx_it_lc_dy_f) tg
on (fs.prod_it_wid = tg.prod_wid and fs.fs_trx_id = tg.tg_trx_id)
) where tg_trx_id is null or prod_wid is null;


-----------------------------------------------
-- Checks on INVENTORY.csv file
-----------------------------------------------
-- Verify initial inventory data before staging it further, check all
columns are populated with expected values (i.e. CTX was properly formed)
select * from W_RTL_INV_IT_LC_DY_FTS

-- Should match the record count from last loaded INVENTORY.csv file
select count(*) from W_RTL_INV_IT_LC_DY_FTS

-- Check that data is making it to target tables after load
select count(*) from W_RTL_INV_IT_LC_DY_F
select count(*) from W_RTL_INV_IT_LC_WK_A

-- inventory validation check after rejected records occur (Support needs to
run this currently, APEX doesn't provide E$ or TMP tables)
SELECT 'E$_W_RTL_INV_IT_LC_DY_TMP','PROD_IT_NUM' DIMM_NAME,PROD_IT_NUM
DIMM_VALUE ,NULL,CHECK_DATE,EFFECTIVE_FROM_DT,EFFECTIVE_TO_DT,'INVALID_PROD_I
T_NUM' INVALID_REASON,NULL,NULL
FROM
(SELECT DISTINCT PROD_IT_NUM,TRUNC(CHECK_DATE) CHECK_DATE, NULL
EFFECTIVE_FROM_DT, NULL EFFECTIVE_TO_DT FROM E$_W_RTL_INV_IT_LC_DY_TMP WHERE
PROD_IT_NUM NOT IN (SELECT PROD_IT_NUM FROM W_PRODUCT_D_RTL_TMP) UNION ALL
SELECT DISTINCT DIMM.PROD_IT_NUM, TRUNC(ERR.CHECK_DATE) CHECK_DATE,
DIMM.SRC_EFF_FROM_DT EFFECTIVE_FROM_DT, DIMM.SRC_EFF_TO_DT EFFECTIVE_TO_DT
from W_PRODUCT_D_RTL_TMP DIMM, E$_W_RTL_INV_IT_LC_DY_TMP ERR WHERE
DIMM.PROD_IT_NUM = ERR.PROD_IT_NUM AND (ERR.DAY_DT > DIMM.SRC_EFF_TO_DT OR
ERR.DAY_DT < DIMM.SRC_EFF_FROM_DT) AND NOT EXISTS (SELECT 1 FROM
W_PRODUCT_D_RTL_TMP DIMM1 WHERE ERR.PROD_IT_NUM = DIMM1.PROD_IT_NUM AND
(ERR.DAY_DT <= DIMM1.SRC_EFF_TO_DT AND ERR.DAY_DT >= DIMM.SRC_EFF_FROM_DT)))
UNION ALL
SELECT 'E$_W_RTL_INV_IT_LC_DY_TMP','ORG_NUM' DIMM_NAME,ORG_NUM
DIMM_VALUE ,NULL,CHECK_DATE,EFFECTIVE_FROM_DT,EFFECTIVE_TO_DT,'INVALID_ORG_DH
_NUM' INVALID_REASON,NULL,NULL
FROM
(SELECT DISTINCT ORG_NUM,TRUNC(CHECK_DATE) CHECK_DATE, NULL
EFFECTIVE_FROM_DT, NULL EFFECTIVE_TO_DT FROM E$_W_RTL_INV_IT_LC_DY_TMP WHERE
ORG_NUM NOT IN (SELECT ORG_NUM FROM W_INT_ORG_DH_RTL_TMP) UNION ALL SELECT
DISTINCT DIMM.ORG_NUM, TRUNC(ERR.CHECK_DATE) CHECK_DATE,
DIMM.EFFECTIVE_FROM_DT EFFECTIVE_FROM_DT, DIMM.EFFECTIVE_TO_DT
EFFECTIVE_TO_DT from W_INT_ORG_DH_RTL_TMP DIMM, E$_W_RTL_INV_IT_LC_DY_TMP
ERR WHERE DIMM.ORG_NUM = ERR.ORG_NUM AND (ERR.DAY_DT > DIMM.EFFECTIVE_TO_DT
OR ERR.DAY_DT < DIMM.EFFECTIVE_FROM_DT) AND NOT EXISTS (SELECT 1 FROM
W_INT_ORG_DH_RTL_TMP DIMM1 WHERE ERR.ORG_NUM = DIMM1.ORG_NUM AND (ERR.DAY_DT
<= DIMM1.EFFECTIVE_TO_DT AND ERR.DAY_DT >= DIMM.EFFECTIVE_FROM_DT)))
```

```
UNION ALL
SELECT 'E$_W_RTL_INV_IT_LC_DY_TMP','ORG_NUM' DIMM_NAME,ORG_NUM
DIMM_VALUE,NULL,CHECK_DATE,EFFECTIVE_FROM_DT,EFFECTIVE_TO_DT,'INVALID_O
RG_NUM' INVALID_REASON,NULL,NULL
FROM
(SELECT DISTINCT ORG_NUM,TRUNC(CHECK_DATE) CHECK_DATE, NULL
EFFECTIVE_FROM_DT, NULL EFFECTIVE_TO_DT FROM E$_W_RTL_INV_IT_LC_DY_TMP
WHERE ORG_NUM NOT IN (SELECT ORG_NUM FROM W_INT_ORG_D_RTL_TMP) UNION
ALL SELECT DISTINCT DIMM.ORG_NUM, TRUNC(ERR.CHECK_DATE) CHECK_DATE,
DIMM.EFFECTIVE_FROM_DT EFFECTIVE_FROM_DT, DIMM.EFFECTIVE_TO_DT
EFFECTIVE_TO_DT from W_INT_ORG_D_RTL_TMP DIMM,
E$_W_RTL_INV_IT_LC_DY_TMP ERR WHERE DIMM.ORG_NUM = ERR.ORG_NUM AND
(ERR.DAY_DT > DIMM.EFFECTIVE_TO_DT OR ERR.DAY_DT <
DIMM.EFFECTIVE_FROM_DT))
UNION ALL
SELECT 'E$_W_RTL_INV_IT_LC_DY_TMP','DAY_DT'
DIMM_NAME,TO_CHAR(DAY_DT,'YYYYMMDD') DIMM_VALUE,
NULL,CHECK_DATE,EFFECTIVE_FROM_DT,EFFECTIVE_TO_DT,'INVALID_DAY_DT'
INVALID_REASON,NULL,NULL
FROM
(SELECT DISTINCT DAY_DT,TRUNC(CHECK_DATE) CHECK_DATE, NULL
EFFECTIVE_FROM_DT, NULL EFFECTIVE_TO_DT FROM E$_W_RTL_INV_IT_LC_DY_TMP
WHERE (DATASOURCE_NUM_ID, DAY_DT)  NOT IN (SELECT
MDAY.DATASOURCE_NUM_ID,MCAL_DAY_DT FROM W_MCAL_DAY_D
MDAY,W_MCAL_CONTEXT_G MTEXT WHERE MDAY.MCAL_CAL_WID=MTEXT.MCAL_CAL_WID
AND MTEXT.ORG_ID IN (Select PARAM_VALUE From C_ODI_PARAM Where
PARAM_NAME = 'ORG_ID' AND SCENARIO_NAME = 'GLOBAL') AND
MTEXT.CALENDAR_ID IN (Select PARAM_VALUE From C_ODI_PARAM Where
PARAM_NAME = 'CALENDAR_ID' AND SCENARIO_NAME = 'GLOBAL')));
```