Oracle® Retail Insights Cloud Service

Implementation Guide

Release 23.2.401.0
F87393-02
October 2023

Oracle Retail Insights Cloud Service Implementation Guide, Release 23.2.401.0

F87393-02

Copyright © 2023, Oracle and/or its affiliates.

Primary Author: Nathan Young

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Send Us Your Comments

Preface

Audience x
Documentation Accessibility x
Customer Support x
Improved Process for Oracle Retail Documentation Corrections xi
Oracle Help Center (docs.oracle.com) xi
Conventions xi

1 Introduction

Business Intelligence and Retail Insights 1-1

2 Setup and Configuration

Initial Configurations 2-1
- Configuration Recommendations 2-7
- Row Delimiter Usage 2-8
- Error Tables 2-8
- Attribute Metadata Configuration 2-10

3 Planning and Flex Fact Configuration

Setting Data Levels 3-1
Preparing Data Files 3-3
Partition Tables 3-4
Loading Plan Data 3-5
Loading Aggregate History 3-6
Applying Configurations 3-6
- Create a Wallet 3-6
- RunPlanningConfigurationScript.sh 3-7

Database Mode (d) 3-8
File Mode (f) 3-8
PlanningConfigurationScript.shScript 3-11

4 Xstore Sales Integration

Data Load Process 4-1
Supported Data Types 4-2
Supported Dimensions 4-3
Intraday Scheduling 4-3

5 Internationalization

Translation 5-1
Multi-Language Setup 5-2
Scenario 1 5-2
Data Scenario 1a 5-2
Data Scenario 1b 5-3
Scenario 2 5-3
Data Scenario 2a 5-3
Scenario 3 5-3

6 Compression and Partitioning

Overview of Compression 6-1
What Compression Does 6-1
Mechanics of Compression 6-2
Compressed Tables and 'CURRENT' Tables 6-2
Coping with Slowly Changing Dimension Type 2 6-3
Fact Close Program (factcloseplp.ksh) 6-3
Fact Open Program (factopenplp.ksh) 6-3
Oracle Table Compression 6-4
Overview of Partitioning Strategies 6-4
Implementing Retail Insights Partitioning 6-5
Setup and Maintenance for Partitioning Retail Insights Compressed Inventory Table 6-5
Implementing Partitioning for Compressed Inventory Table 6-6
Partitioning Automation Implementation 6-7
Partitioning Automation Operation 6-8
How Oracle Implements Partitions 6-8
Summary 6-9

7 Performance

Key Factors in Performance 7-1
- Purging and Archiving Strategy 7-1
- Flexible Aggregates 7-2
- ETL Programs Performance 7-3
  - Setting ETL Program Multi-threading 7-3
  - ODI Configuration 7-4
  - ETL Batch Scheduling 7-4
  - Additional Considerations 7-5
- Report Design 7-5
- Additional Factors 7-5
- Partitioning Strategy 7-6
- Data Base Configuration 7-6
- Adequate Hardware Resources 7-6

Leading Practices 7-6
- Customizations 7-6
- ODI Best Practices 7-7
- Oracle Analytics Best Practices 7-7
- Batch Schedule Best Practices 7-7
  - Automation 7-7
  - Recoverability 7-7
- Retail Insights Loading Batch Execution Catch-Up 7-7
  - High Availability 7-8
  - Batch Efficiency 7-9

Aggregates List 7-9

8 Retail Insights Universal Adapter

Overview of Retail Insights Universal Adapter Framework 8-1
- Benefits 8-2

Universal Adapter Installation and Configuration 8-2

Universal Adapter Execution 8-3

9 OAS Cache Usage

Step1: Set up OAS Caching 9-1

Step 2: Setup Parameters for OAS Caching in ra.env 9-3

Step3: Delete and Recreate the Clear Cache Agent 9-5

Step 4: Clearing Cache using the Clear Cache Script 9-6

10 Frequently Asked Questions

List of Figures

1-1	Data Element Relationships	1-2
3-1	Properties.txt File	3-9
6-1	Retail Insights Partitioning Options	6-5
8-1	RDE to Retail Insights Staging Data Flow	8-1
8-2	Moving Third Party Extracts into Retail Insights Staging Tables	8-2
9-1	Target Navigation Menu	9-1
9-2	Overview Tab	9-2
9-3	Configuration Tab	9-2
9-4	Lock and Edit	9-2
9-5	Target Navigation Menu	9-3
9-6	Availability Tab - Non-Clustered Environment	9-4
9-7	Availability Tab - Clustered Environment	9-4
9-8	Delete the Agent	9-5
9-9	Create Agent	9-5
9-10	Choose Delivery Content	9-6
9-11	Save as Cache Clear Agent	9-6

List of Tables

2-1	C_ODI_PARAM Initial Setup	0
2-2	C_ODI_PARAM RDE Configurations	0
2-3	Configuration Recommendations	0
2-4	Core Data Load Error Tables	0
2-5	W_RTL_UDA_METADATA_G Interface Columns	0
3-1	C_ODI_PARAM Planning and Flex Fact Parameters	3-1
3-2	C_ODI_PARAM Planning and Flex Fact Values	3-2
3-3	Dataset Jobs	0
4-1	Transaction Usage Details	0
4-2	Supported Dimensions	0
6-1	Compressed Tables and CURRENT Tables	6-3
7-1	Retail Insights Aggregates	7-9

Send Us Your Comments

Oracle Retail Insights Implementation Guide, Release 23.2.401.0

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

The Oracle Retail Insights Cloud Service Implementation Guide provides detailed information useful for implementing the application. It helps you to view and understand the behind-the-scenes processing of the application.

Audience

The Implementation Guide is intended for Oracle Retail Insights application integrators and implementation staff.

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

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

Oracle Retail product documentation is available on the following web site:

https://docs.oracle.com/en/industries/retail/index.html

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Help Center (docs.oracle.com)

Oracle Retail product documentation is available on the following web site:

https://docs.oracle.com/en/industries/retail/index.html

(Data Model documents can be obtained through My Oracle Support.)

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction

Retail Insights offers cloud-based rich business intelligence solution to retail industry users. Retail Insights is built on top of the latest Oracle technology stack and utilizes Oracle Data Integrator (ODI) for extracting, transforming, and loading (ETL) the data and Oracle Analytics Server (OAS) for end user reporting and analysis needs.

Retail Insights architecture is designed to meet the retail industry's business intelligence needs in both program and report performance.

The main characteristics of the Retail Insights product are:

- Rich Reporting Capabilities: Retail Insights offers report creation capabilities in two different flavors: Historical (As Was) and Current (As Is) in same environment. Packaged reports are provided as reference examples for users to create their own customized reports according to their needs.
- Comprehensive Solution: Retail Insights includes an end-to-end solution for reporting and BI needs of the retailer by providing data integration with source applications, transforming and loading the fact and dimension data, rolling up the data for improved query performance, Web-based graphical user interface (GUI) for report creation, shell scripts for setting up the batch schedule, and an automated installer by following business intelligence best practices.
- Performant ETL Code: Retail Insights data processing tool, ODI, offers high performance for the database batch processes on Oracle database.
- Performant Reports: Retail Insights metadata is built using Oracle Analytics and are designed to work in complex reporting scenarios.
- Robust Data Model: Retail Insights data model is designed for supporting a retailers' data needs in a business intelligence environment. Data model elements are designed to work with Oracle Analytics architecture.
- Packaged Integrations: Retail Insights data model is leveraged for many down-stream applications such as the Retail AI Foundation Cloud Services with built-in integrations and data flows.

Business Intelligence and Retail Insights

This section briefly explains the fundamentals of business intelligence and data warehousing in general. It is important to understand the overall architecture and data flow for implementing Retail Insights.

Business intelligence includes the processes, methods, and technologies adopted by organizations to answer complex business questions and for building comprehensive decision support systems. These systems help organizations in maintaining secure, conformed, and highly available data for all levels of users from top executives who make decisions based on corporate level information to managers/analysts who analyze their area and take actions based on the information.

Business intelligence is built using several processes and applications that maintain these processes by adopting latest tools and technologies. One of the main components of

business intelligence is a data warehouse. A data warehouse is the repository that stores the data extracted from several source systems and modelled to perform for data loading, reporting, and ad-hoc analysis needs.

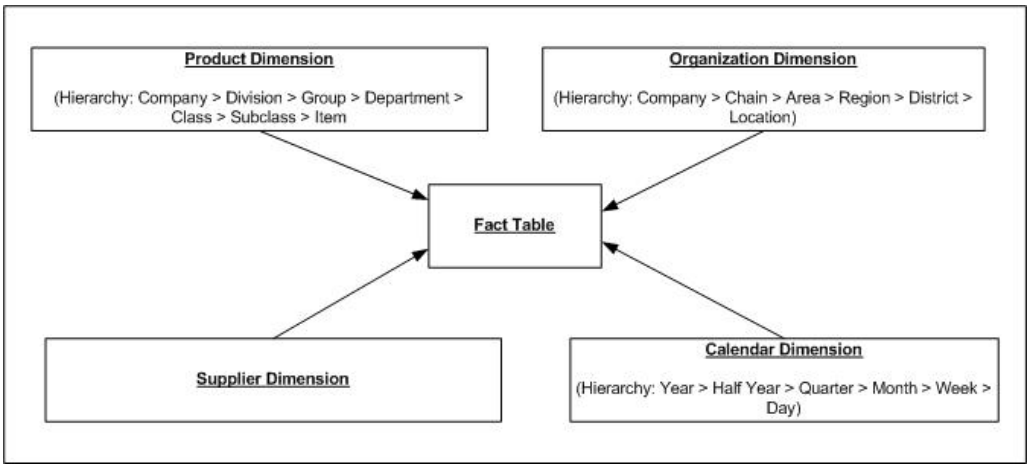
Retail Insights uses sophisticated techniques to populate the data warehouse. Explained in greater detail throughout this guide, these techniques include taking the data provided by Oracle Retail Data Extractor (RDE) and then rapidly transforming that data and loading it into the data warehouse. Techniques used to load data into the warehouse vary depending upon whether the data consists of facts or dimensions.

There are several fact and dimension tables in the subject areas available in Retail Insights. Some examples of subject areas that exist in Retail Insights include Sales, Inventory Position, and Base Cost. Each subject area has its own data mart to support reporting and analytic needs. At the center of each data mart is fact data (note that fact data here corresponds to both base fact data and aggregated data). Facts are the transactions that occur in your data warehouse's source systems, such as RMS. You may want to look at sales transaction facts, inventory stock count facts at stores or warehouses, or inventory movement facts.

Facts have little meaning by themselves because they are usually just values (for example, six sales at a store, 15 items left at a warehouse, or 300 items transferred). What gives fact data true meaning is the intersection of dimensions in which facts exist. In other words, six sales on Wednesday at store B, or 15 dishwashers in stock last Monday at the Chicago warehouse, or 300 blouses transferred during the last week in February from the St. Louis warehouse to the Denver warehouse. Dimension data, therefore, exists in the data warehouse to serve as reference data to facts.

The following diagram illustrates data elements of a generic data mart and their inter-relationships:

Figure 1-1 Data Element Relationships



2

Setup and Configuration

The Retail Insights application is a part of the Retail Analytics and Planning solutions and much of the implementation steps have been combined into a single platform-level document referred to as the Retail Analytics and Planning Implementation Guide. For general guidance on implementing RI or AI Foundation applications, first refer to that document. The chapters of this guide supplement the platform documents with RI-specific details as needed.

Initial Configurations

After performing all instructions for initial environment configuration in the RAP Implementation Guide, you may also want to configure Retail Insights-specific parameters that will impact your data loading and conversion processes. Review the table below for a complete list of these parameters.

Additionally, the Retail Data Extractor (RDE) tool has some configurations specifically for controlling the ETL logic between RMFCS and RI. These settings do not apply if you are implementing the Retail Insights without RMFCS. The RDE settings are available from the Control Center in the same C_ODI_PARAM_VW table used for RI.

Table 2-1 C_ODI_PARAM Initial Setup

Scenario	Parameter	Usage
GLOBAL	RI_UA_ROW_DELIMITER	Change the row ending characters for all data files sent into RI, both from RDE and external sources. If the default value of "\n" may occur in text strings, a custom delimiter MUST be set before loading data.
GLOBAL	LANGUAGE_CODE	Default language code used by the system to load data. Do not change unless your source systems are also using a non-English default language.
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. Default=91 days.
GLOBAL	RI_CLOSED_PO_HIST_DAYS	The number of days to retain closed purchase orders on the daily positional snapshots. Default=30 days.
GLOBAL	RI_PART_DDL_CNT_LIMIT	Maximum number of partitions to create during the initial setup run, recommended value is 100000. The average initial setup of the calendar may need 50-60,000 partitions.
GLOBAL	RA_CLR_LEVEL	Disables the mapping of clearance event IDs to clearance inventory updates, set to N to disable. Disabling may improve batch performance.

Table 2-1 (Cont.) C_ODI_PARAM Initial Setup

Scenario	Parameter	Usage
GLOBAL	ANCHOR_TO_YEARS	Number of years of data to consider for Same Stores method of Comparable Store reporting. Set to 3 if LLY is required.
GLOBAL	SAME_STORES	Enable or disable Same Stores method of Comparable Store reporting.
GLOBAL	GIFT_CARD_TENDER_TYPE_ID	The tender type ID associated with gift cards in RMS, for gift card fact usage.
GLOBAL	CLSTR_GRP_TYPE	Controls the type of data loaded to the Cluster interfaces, either CLSTR for store clusters or PRICE_ZONE for price zones.
GLOBAL	ITEM_CFA_VISIBILTY LOC_CFA_VISIBILTY SUPS_CFA_VISIBILTY ITEM_LOC_CFA_VISIBILTY MCAL_DAY_CFA_VISIBILTY	Controls access to CFAS attributes in OAS reporting, set to 0 to enable, 1 to disable.
GLOBAL	LY_SHIFT_TYPE	Controls the default LY calendar mapping used, accepts values in UNSHIFT, SHIFT, or GUNSHIFT. The first two are fiscal calendar variations, while the third is Gregorian calendar.
GLOBAL	ITEM_GRP1_IS_INCREMENTAL	Enable incremental processing of the W_RTL_ITEM_GRP1_D table, which is used for product attributes, UDAs, and item lists. Should be set to 'Y' in production environments once initial loading is done.
GLOBAL	RTVR_REASON_CAT_CODE	RMS code type for RTV reason codes.
GLOBAL	RI_TRX_COUNT_RECLASS_IND	Set to 'Y' to enable reclassification processing of transaction count aggregates. This may greatly impact batch runtimes so it should not be enabled unless required.
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, in order 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 in order 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.

Table 2-1 (Cont.) C_ODI_PARAM Initial Setup

Scenario	Parameter	Usage
GLOBAL	RI_PROD_DS_MANDATORY_IND	Set to 'Y' to require input data on the Product hierarchy interface in order 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	CURRENCY_CODE	Set the default currency code to use when loading CSV-based fact data files if none are provided on the files themselves. Defaults to 'USD'.
GLOBAL	RI_EXCLUDE_VAT_IND	Determines if special Sales metrics are displayed in OAS for "Sales excluding VAT". These metrics subtract the tax amounts from the sales values under the assumption tax is always included in the inputs, but should not be shown in reporting. Set to '0' to show the metrics, or '1' to hide them.
GLOBAL	START_OF_YEAR_MONTH	The name of the Gregorian month associated with the first fiscal period in your business calendar. For example, if your fiscal year starts 06-FEB-22 then set this to FEBRUARY. This will be used to display month names in RI reporting on the fiscal calendar. Default = JANUARY
GLOBAL	HIST_ZIP_FILE	Change the default name for the ZIP file package used by the history file load process. Default=RAP_DATA_HIST.zip
GLOBAL	RI_AGG_FULL_LOAD_TYPE	Controls date extending behavior for aggregation utility. Should be one of (F, FS, FE, NA). Default = FS
GLOBAL	RI_ITEM_REUSE_IND	Enable or disable the ability to re-use item numbers over time to represent entirely new items. Also enables retention of existing items for a number of days, so that if an item drops and reappears quickly, it is not considered a new item and will continue to use the existing records. Set to 'Y' to enable. If this is not enabled, items that are dropped from the product interface are immediately closed and deactivated and cannot be re-opened. Default = N

Table 2-1 (Cont.) C_ODI_PARAM Initial Setup

Scenario	Parameter	Usage
GLOBAL	RI_ITEM_REUSE_AFTER_DAYS	The number of days between when an item is deleted and when it's allowed to appear as a new item having the same ID. This will trigger the old version of the item to be archived in the data warehouse using an alternate key, so the new version of the item is treated as completely new. For example, setting this to 5 days means that an item can be dropped/ deleted and after 5 days, when the same items come again it will be treated as a brand-new item. If the item re-appears in the data before 5 days has passed, it will be treated as the same item as before and the existing item data remains active. Default = 0
SIL_DAYDIMENSION	END_DT	End date for generating the system calendar (this is different from the fiscal calendar). Set at least 6 months beyond the end of the fiscal calendar.
SIL_DAYDIMENSION	START_DT	Start date for generating the system calendar (this is different from the fiscal calendar). Set at least 6 months before the start of the fiscal calendar.
SIL_DAYDIMENSION	WEEK_START_DT_VAL	Starting day of the week for the system calendar (1 = Sunday).
SIL_RETAILINVPOSITIONFACT	RI_INVAGE_REQ_IND	Disables calculation of first/last receipt dates and inventory age measures. Disabling may improve batch performance.
SIL_RETAILINVPOSITIONFACT	RI_PRES_STOCK_IND	Disables usage of replenishment data for presentation stock to calculate inventory availability measures Disabling may improve batch performance.
SIL_RETAILINVPOSITIONFACT	RI_BOH_SEEDING_IND	Disables the creation of initial beginning-on-hand records so analytics have a non-null starting value in the first week. Disabling may improve batch performance.
SIL_RETAILINVPOSITIONFACT	RI_MOVE_TO_CLR_IND	Disables calculation of move-to-clearance inventory measures when an item/location goes into or out of clearance status. Disabling may improve batch performance.
SIL_RETAILINVPOSITIONFACT	RI_MULTI_CURRENCY_IND	Disables recalculation of primary currency amounts if you are only using a single currency. Disabling may improve batch performance.
SIL_ITEMDIMENSION	IS_INCREMENTAL	Controls if the product dimension is full snapshot or incremental changes only for the daily load.
SIL_RETAILITEMCFADIMENSION	IS_INCREMENTAL	Controls if the product CFAS dimension is full snapshot or incremental changes only for the daily load.

Table 2-1 (Cont.) C_ODI_PARAM Initial Setup

Scenario	Parameter	Usage
SIL_RETAILITEMLOCCFADIMENSION	IS_INCREMENTAL	Controls if the product loc CFAS dimension is full snapshot or incremental changes only for the daily load.
SIL_RETAILLOCATIONCFADIMENSION	IS_INCREMENTAL	Controls if the location CFAS dimension is full snapshot or incremental changes only for the daily load.
SIL_RETAILSUPPCFADIMENSION	IS_INCREMENTAL	Controls if the supplier CFAS dimension is full snapshot or incremental changes only for the daily load.
SIL_RETAILSUBSTITUTEITEMDIMENSION	IS_INCREMENTAL	Controls if the substitute item dimension is full snapshot or incremental changes only for the daily load.
SIL_RETAILPROMOTIONDIMENSION	RI_INCREMENTAL_IND	Controls if the promotion dimension is full snapshot or incremental changes only for the daily load.
SIL_SEEDEMPLOYEEDIMENSION	RI_MIS_CASHIER_REQ_IND	Seed missing Cashier IDs from sales fact to Employee dimension.
SIL_SEEDCOHEADDIMENSION	RI_MISSLS_COHEAD_REQ_IND	Seed missing customer order (CO) head IDs from sales fact to CO Dimension.
SIL_SEEDCOLINEDIMENSION	RI_MIS_COLINE_REQ_IND	Seed missing customer order (CO) line IDs from sales fact to CO Dimension.
SIL_SEEDCOUPONDIMENSION	RI_MIS_COUPON_REQ_IND	Seed missing coupon IDs from sales discount fact to Coupon dimension.
SIL_SEEDCUSTOMERDIMENSION	RI_MIS_CUSTOMER_REQ_IND	Seed missing customer IDs from sales fact to Customer dimension.
SIL_SEEDCUSTOMERLOYALTYAWARDACCOUNTDIMENSION	RI_SEED_AWARD_ACCOUNT_IND	Seed missing award account IDs from loyalty fact to Award Account dimension.
SIL_SEEDDISCOUNTTYPEDIMENSION	RI_MIS_DISC_TYPE_REQ_IND	Seed missing discount type codes from sales discount fact to Discount Type dimension.
SIL_SEEDPROMOTIONDIMENSION	RI_MISSLS_PROMO_REQ_IND	Seed missing promotions from the sales promo fact to the Promotion dimension.
SIL_SEEDSTOCKCOUNTDIMENSION	RI_MIS_STOCK_CNT_REQ_IND	Seeds missing stock counts from the systemic (RMS) stock count fact to the stock count dimension.
SIL_RETAILSALESPROMOTIONTRANSACTIONFACT	RI_EXT_PROMO_COL	Select a reference column on the sales fact which represents an External Promotion. This will be joined with W_RTL_PROMO_EXT_DS interface to load externally sourced promotion sales into RI that don't exist in Pricing CS, and treat them as valid promo sales.
SIL_RETAIL_SALESPOS_DATA_HANDLING	PURGE_POSXML_DAYS	Number of days to keep POS sales logs in raw XML format. This data is only intended for internal debugging/error resolution, as POS sales are only shown in reporting for the current date.

Table 2-2 C_ODI_PARAM RDE Configurations

Scenario	Parameter	Usage
GLOBAL	RDE_UA_ROW_DELIMITER	Default row delimiter on incoming data files (both RMS and external).
GLOBAL	RI_UA_ROW_DELIMITER	Default row delimiter on data files going out to RI, must match with RI incoming UA delimiter value.
GLOBAL	RPM_PROMO_EVENT_LEVEL	Enable (set to 'Y') if using legacy RPM on-premise functionality.
GLOBAL	RETURN_REASON_CAT_CODE	RMS code type for customer return reason codes.
GLOBAL	RTVR_REASON_CAT_CODE	RMS code type for RTV reason codes.
GLOBAL	WHOLESALE_CHANNEL	Identify if there is a wholesale channel setup in RMS.
GLOBAL	ITEM_GRP1_IS_INCREMENTAL	Controls if item attributes are incremental or full snapshot on the daily load. Must be in sync with the same-named RI parameter.
GLOBAL	RMS_VERS_CHECK	Controls the behavior of code that is linked to a specific RMS version.
GLOBAL	RA_INV_WAC_IND	Controls the inventory cost calculation in RDE. When set to 'Y' it will use Weighted Avg Cost (WAC) as the item cost for all items. When set to 'N' it will dynamically load RMFCS valuation methods set per department or item and apply them, choosing from avg cost, unit cost, and retail-based cost.
GLOBAL	RA_INV_TAX_IND	Controls the calculation and removal of tax amounts from retail valuation of stock on hand and on-order amounts. When set to 'N', only simple VAT (SVAT) calculations are supported and non-VAT items are left as-is. When set to 'Y', the system dynamically loads RMFCS global tax and VAT information and applies it by item/loc.
GLOBAL	RA_SLS_TAX_IND	Controls the calculation and removal of tax amounts from retail valuation of sales. When set to 'N', VAT rates are included in the sales retail amounts (this is how it comes from Sales Audit). When set to 'Y' the VAT is removed from the retail amounts. Profit is always VAT-exclusive in either case, this applies to columns based on the total selling retail like SLS_AMT and RET_AMT.
SDE_RETAILINVRECEIPTSFACT	VWH_NO_ALC_RCPTS	Specify a type of warehouse that cannot receive allocations in the feed to RI. Converts the allocs to normal transfer receipts. Uses codes from VWH_TYPE column in RMFCS.
SDE_RETAILINVRECEIPTSFACT	STORE_NO_ALC_RCPTS	Specify a type of store that cannot receive allocations in the feed to RI. Converts the allocs to normal transfer receipts.

Table 2-2 (Cont.) C_ODI_PARAM RDE Configurations

Scenario	Parameter	Usage
SDE_RETAILITEMDIMENSION	IS_INCREMENTAL	Controls if the product dimension is full snapshot or incremental changes only for the daily load.
SDE_RETAILITEMLOCATIONRANGEDIMENSION	IS_INCREMENTAL	Controls if the product location range dimension is full snapshot or incremental changes only for the daily load.
SDE_RETAILITEMSUPPLIERDIMENSION	IS_INCREMENTAL	Controls if the supplier-item dimension is full snapshot or incremental changes only for the daily load.
SDE_RETAILSUBSTITUTEITEMDIMENSION	IS_INCREMENTAL	Controls if the substitute-item dimension is full snapshot or incremental changes only for the daily load.
SDE_RETAILITEMLOCATIONDIMENSION	IS_INCREMENTAL	Controls if the product location attr dimension is full snapshot or incremental changes only for the daily load.
SDE_RETAILITEMLOCCFADIMENSION	IS_INCREMENTAL	Controls if the product loc CFAS dimension is full snapshot or incremental changes only for the daily load.
SDE_RETAILSUPPCFADIMENSION	IS_INCREMENTAL	Controls if the supplier CFAS dimension is full snapshot or incremental changes only for the daily load.
SDE_RETAILLOCATIONCFADIMENSION	IS_INCREMENTAL	Controls if the location CFAS dimension is full snapshot or incremental changes only for the daily load.
SDE_RETAILITEMCFADIMENSION	IS_INCREMENTAL	Controls if the product CFAS dimension is full snapshot or incremental changes only for the daily load.

Configuration Recommendations

Table 2-3 Configuration Recommendations

Scenario	Details
First Time Dimension Loads (Full vs. Incremental)	When first loading data into RI, you will need to ensure all IS_INCREMENTAL flags are set to a value of N, meaning the programs will expect full snapshots of your data and not deltas only.
First Time Dimension Loads (Seeding)	When loading fact data for the first time, you will want to verify the seeding indicators are mostly set to a value of Y, meaning that if any fact record has an unknown identifier, we will try to create the record for it to avoid rejected data or failures. For example, RI_MIS_COHEAD_REQ_IND must be set to Y to auto-seed customer order IDs that may appear on your sales transaction history.

Table 2-3 (Cont.) Configuration Recommendations

Scenario	Details
Inventory History Loads	When loading inventory history, you should disable (set to N) all configuration options that you do not need in the final dataset, in order to improve the load performance. For example, RI_BOH_SEEDING_IND should be disabled if you don't need RI to create zero-balance BOH records, as these records are purely for reporting purposes to show a zero instead of null values in initial BOH positions.
Daily/Weekly Data Loads (Full vs. Incremental)	Once you are done with initial data loads, you may want to switch interfaces from full snapshot to incremental. Update the IS_INCREMENTAL parameters to Y at this time to start accepting regular delta files. If you are loading data from RMFCS you must synchronize this between programs to avoid failures in the batch (for example, if you change RDE to send incremental attributes data, but RI expects full snapshot, then the RI batch will deactivate all records which don't come on the file).
Report Behaviors	Several parameters in C_ODI_PARAM actually affect reporting behaviors and need to be configured properly before allowing end-users into the system. This includes: Comp Store handling, CFAS attribute visibility, and LY calendar shift type.

Row Delimiter Usage

The configuration setting in C_ODI_PARAM for RI_UA_ROW_DELIMITER can be very helpful when integrating data from RMFCS that may contain traditional line-ending characters like \n or \r\n (the UNIX and Windows default line endings). When line-endings are encountered within a record that match the actual line-ending character on the data file rows, RI will fail to process those records because of the additional row breaks. For example, users may accidently copy and paste a string of text into RMFCS that includes these line-ending characters but have no knowledge of it, because the characters will be invisible when displayed back in the RMFCS UI.

With this RI_UA_ROW_DELIMITER parameter, you have the option to add additional characters to the RDE/RI flat file row delimiter, effectively allowing the system to process mid-line breaks without failing. A common sequence used is ~~\n instead of the default \n. This will cause RDE to append the ~~ symbols to every line of output data, and likewise RI will only accept line breaks if the same sequence is found. Once the parameter is set on both RDE and RI, the effect on the data is entirely transparent to the end user, as the extra characters will be ignored after the data is pulled into the RI database.

Also note that this parameter can apply to both DAT and CSV files. This means that by default, CSV files which don't come from RMFCS should still use the line-ending character set on this parameter (i.e. UNIX line endings). If your files will have Windows line-endings, then you may want to change RI to use \r\n for this parameter. If you are using a version of RI that allows line-endings to be set on the Context Files, then this does not apply as the context files will override the system setting.

Error Tables

The following table describes database tables you may need to query during the initial data load process to identify and resolve rejected records in RI (using Data Visualizer to access the database, as described in the RAP Implementation Guide). Note that error tables do not exist unless created as part of the rejection process. The list below describes the most commonly used rejection tables, but all fact interfaces to RI have the ability to generate one using a similar naming scheme.

Table 2-4 Core Data Load Error Tables

Table Name	Usage
E$_W_RTL_BCOST_IT_LC_DY_TMP E$_W_RTL_NCOST_IT_LC_DY_TMP E$_W_RTL_PRICE_IT_LC_DY_TMP	Rejection of base cost, net cost, and price records. The most common rejection reason is due to inactive items, locations, or suppliers in the dimension data. Closed items or locations should stop getting data on these interfaces but it can happen that the source system tries to send an update for them.
E$_W_RTL_CUST_LYL_AWD_TRX_DY_T E$_W_RTL_CUST_LYL_TRX_LC_DY_TM	Rejection of customer loyalty transaction records. The most common rejection reason is due to inactive or missing customers or loyalty account records.
E$_W_RTL_FLEXFACT1_TMP E$_W_RTL_FLEXFACT2_TMP E$_W_RTL_FLEXFACT3_TMP E$_W_RTL_FLEXFACT4_TMP	Rejection of flexible fact records. The most common reason for rejections is due to dimension identifiers on the incoming file not aligning with the configured data levels of the flex fact (such as department identifiers in a file intended for class level data).
E$_W_RTL_INVADJ_IT_LC_DY_TMP E$_W_RTL_INVRC_IT_LC_DY_TMP E$_W_RTL_INVRECLASS_IT_LC_DY_T E$_W_RTL_INVRTV_IT_LC_DY_TMP E$_W_RTL_INVTSF_IT_LC_DY_TMP	Rejection if various inventory transaction records. The most common reason for rejections is due to invalid or missing status codes, reason codes, or product/location records in the dimension data.
E$_W_RTL_INV_IT_LC_DY_TMP E$_W_RTL_INV_IT_LC_DY_TMP1	Rejection of inventory position records. Positional data may be rejected if an item or location record is not active on the effective date of the fact record. The second table is only used for history load rejections.
E$_W_RTL_PLAN1_PROD1_LC1_T1_TMP E$_W_RTL_PLAN2_PROD2_LC2_T2_TMP E$_W_RTL_PLAN3_PROD3_LC3_T3_TMP E$_W_RTL_PLAN4_PROD4_LC4_T4_TMP E$_W_RTL_PLAN5_PROD5_LC5_T5_TMP E$_W_RTL_PLANFC_PROD1_LC1_T1_T E$_W_RTL_PLANFC_PROD2_LC2_T2_T	Rejection of planning and forecast data. The most common reason for rejections is due to dimension identifiers on the incoming file not aligning with the configured data levels of the fact (such as department identifiers in a file intended for class level data).
E$_W_RTL_SLSDSC_TRX_IT_LC_DY_T	Rejection of sales discount data. The most common reason for rejections is due to invalid/missing discount type codes, promotion codes, or inactive item/locations on the date the transaction occurred.

Table 2-4 (Cont.) Core Data Load Error Tables

Table Name	Usage
E$_W_RTL_SLS_TRX_IT_LC_DY_TMP	Rejection of sales transaction data. Sales transactions have a large number of primary keys and could be rejected due to invalid/missing dimensions on any of them (employee, customer order number, item, location, date, time-of-day, etc.)
E$_W_RTL_TRX_TNDR_LC_DY_TMP	Rejection of sales tender data. The most common reason for rejections is due to invalid/missing tender type codes or inactive locations on the date the transaction occurred.

Attribute Metadata Configuration

If you plan to use User-Defined Attributes (UDAs) in RI reports, then you will also want to provide a UDA configuration file to setup your most important reporting attributes. When UDA data first comes into RI, it is held in a raw row-based format where each row is an item/attribute value pair. From that data, RI can take up to 50 attribute groups and pivot them into a column-based format that is better suited to most reporting needs. Only attributes which are pivoted in this manner can be displayed side-by-side in reports as named columns. Unpivoted attributes have to pull from the larger row-based dataset which can take a significant amount of time to return results and cannot be displayed with more than one attribute group at a time.

The interface file for this configuration is W_RTL_UDA_METADATA_G.dat. It is a full load interface, meaning it should be sent nightly in the batch or the data will be dropped. If you do not have the ability to send it nightly, you can instead disable the job in POM after the file is loaded once. The file uses the legacy RI data format with pipe delimiters and Unix line endings. All columns on the interface should be provided in the file. The table below describes the fields in the interface.

Table 2-5 W_RTL_UDA_METADATA_G Interface Columns

Field Name	Usage
ATTR_NAME	The attribute group ID for the UDA you want to place in a pivoted column.
SOURCE	Reference code for the source system providing the attributes data. Is not used at this time and could be hard-coded as “RMS”.
PHYSICAL_COL_NAME	The target column in RI where this UDA will be stored. Column names start from UDA_ATTR01_NAME and go through UDA_ATTR50_NAME.
TABLE_NAME	Reference code for the RI data table. Currently should be hard-coded as “W_PRODUCT_ATTR_D”
DESCRIPTION	Descriptive value for the attribute group specified on ATTR_NAME.

Table 2-5 (Cont.) W_RTL_UDA_METADATA_G Interface Columns

Field Name	Usage
DATA_TYPE	Not used at this type – leave blank in the file
DATASOURCE_NUM_ID	Hard-code as “1”
INTEGRATION_ID	Set equal to the ATTR_NAME
TENANT_ID	Not used at this type – leave blank in the file
X_CUSTOM	Not used at this type – leave blank in the file

The data below shows example records for five UDA groups which will be pivoted into the first five columns in the attributes table.

- 45|RMS|UDA_ATTR01_NAME|W_PRODUCT_ATTR_D|Color Family||1|45||
- 61|RMS|UDA_ATTR02_NAME|W_PRODUCT_ATTR_D|Silhouette||1|61||
- 7|RMS|UDA_ATTR03_NAME|W_PRODUCT_ATTR_D|Material||1|7||
- 18|RMS|UDA_ATTR04_NAME|W_PRODUCT_ATTR_D|Print||1|18||
- 29|RMS|UDA_ATTR05_NAME|W_PRODUCT_ATTR_D|Design||1|29||

Once this file is loaded into RI, the job which loads the UDA data in POM is W_RTL_PRODUCT_ATTR_UDA_D_JOB. This job will populate the target table W_RTL_PRODUCT_ATTR_UDA_D. If you are not sending the metadata file every night in batch, then after the load is successful you should disable the job W_RTL_UDA_METADATA_G_JOB and restart your POM schedule so it will take effect for the next batch.

The pivoted UDA data is accessible using Item dimension attributes. By default, these are named like “Item UDA ID 1” and “Item UDA Desc 1”. You may rename the attributes to better match their contents using Resource Bundle Customization screens in Retail Home. You may add one or more of these attributes into your Item level reporting, or use them to aggregate item level data to the UDA level of summarization.

3
Planning and Flex Fact Configuration

Retail Insights supports Merchandise Financial Planning (MFP) data on four separate interfaces, referred to as PLAN1 through PLAN4 tables, Assortment Planning (AP) data on a fifth interface called PLAN5, as well as two IPO Cloud Service-Demand Forecasting forecast interfaces (PLANFC1 and PLANFC2) and four flexible fact interfaces (FLEXFACT1 to FLEXFACT4) for any other external data. All of these tables have configurable data levels to align with the hierarchy levels used in MFP, IPO Cloud Service-Demand Forecasting, or other source systems, but they must be configured prior to using them. The following section describes how to configure RI flexible interfaces for first-time use.

Setting Data Levels

Perform the following procedure to update the expected data levels for the planning interfaces. These levels determine which values the primary key columns are joined with inside Retail Insights (for example, is your plan at Department level or Subclass level). Each interface can be configured with a different set of levels.

1. Access the Control & Tactical Center to make updates to the C_ODI_PARAM_VW table.
2. Search for parameter names in the list below, depending on which interfaces you wish to use:

Table 3-1 C_ODI_PARAM Planning and Flex Fact Parameters

Planning Facts	Flexible Facts	Planning Forecast Facts
RI_PLAN1_CAL_LEVEL	RI_FLEXFACT1_CAL_LEVEL	RI_PLANFC1_CAL_LEVEL
RI_PLAN1_ORG_LEVEL	RI_FLEXFACT1_ORG_LEVEL	RI_PLANFC1_ORG_LEVEL
RI_PLAN1_PROD_LEVEL	RI_FLEXFACT1_PROD_LEVEL	RI_PLANFC1_PROD_LEVEL
RI_PLAN1_SUPP_LEVEL	RI_FLEXFACT1_SUPP_LEVEL	RI_PLANFC1_SUPP_LEVEL
RI_PLAN1_ATTR_LEVEL	RI_FLEXFACT1_ATTR_LEVEL	RI_PLANFC1_ATTR_LEVEL
RI_PLAN2_CAL_LEVEL	RI_FLEXFACT2_CAL_LEVEL	RI_PLANFC2_CAL_LEVEL
RI_PLAN2_ORG_LEVEL	RI_FLEXFACT2_ORG_LEVEL	RI_PLANFC2_ORG_LEVEL
RI_PLAN2_PROD_LEVEL	RI_FLEXFACT2_PROD_LEVEL	RI_PLANFC2_PROD_LEVEL
RI_PLAN2_SUPP_LEVEL	RI_FLEXFACT2_SUPP_LEVEL	RI_PLANFC2_SUPP_LEVEL
RI_PLAN2_ATTR_LEVEL	RI_FLEXFACT2_ATTR_LEVEL	RI_PLANFC2_ATTR_LEVEL

Table 3-1 (Cont.) C_ODI_PARAM Planning and Flex Fact Parameters

Planning Facts	Flexible Facts	Planning Forecast Facts
RI_PLAN3_CAL_LEVEL	RI_FLEXFACT3_CAL_LEVEL	
RI_PLAN3_ORG_LEVEL	RI_FLEXFACT3_ORG_LEVEL	
RI_PLAN3_PROD_LEVEL	RI_FLEXFACT3_PROD_LEVEL	
RI_PLAN3_SUPP_LEVEL	RI_FLEXFACT3_SUPP_LEVEL	
RI_PLAN3_ATTR_LEVEL	RI_FLEXFACT3_ATTR_LEVEL	
RI_PLAN4_CAL_LEVEL	RI_FLEXFACT4_CAL_LEVEL	
RI_PLAN4_ORG_LEVEL	RI_FLEXFACT4_ORG_LEVEL	
RI_PLAN4_PROD_LEVEL	RI_FLEXFACT4_PROD_LEVEL	
RI_PLAN4_SUPP_LEVEL	RI_FLEXFACT4_SUPP_LEVEL	
RI_PLAN4_ATTR_LEVEL	RI_FLEXFACT4_ATTR_LEVEL	
RI_PLAN5_CAL_LEVEL		
RI_PLAN5_ORG_LEVEL		
RI_PLAN5_PROD_LEVEL		
RI_PLAN5_SUPP_LEVEL		
RI_PLAN5_ATTR_LEVEL		

3. Modify the parameters as needed using the following list of supported values. A value of ALL means the column should be set to a default value of -1 on the data file and it will not be joined with any other dimension in RI during ETL. A value of FLEX# indicates an alternate hierarchy level will be used, such as FLEX1 matching data on the W_PRODUCT_FLEX_D.FLEX1_CHAR_VALUE field. Usage of flex fields assumes that this is the same level also used in Planning applications for the plan data.

Table 3-2 C_ODI_PARAM Planning and Flex Fact Values

Product (PROD)	Organization (ORG)	Calendar (CAL)	Supplier (SUPP)	Attribute (ATTR)
CMP	COMPANY	YEAR	SUPPLIER	C (for Color)
DIV	CHAIN	HALFYEAR	PARENT_SUPPLIER	B (for Brand)
GRP	AREA	QUARTER	ALL	FLAVOR
DEPT	REGION	PERIOD		FABRIC
CLS	DISTRICT	WEEK		SCENT
SBC	LOCATION	DAY		SIZE

Table 3-2 (Cont.) C_ODI_PARAM Planning and Flex Fact Values

Product (PROD)	Organization (ORG)	Calendar (CAL)	Supplier (SUPP)	Attribute (ATTR)
ITEM	CHANNEL	GREGORIANYEAR		STYLE
ALL	PLANNING_CHANNEL	GREGORIANQUARTER		ALL
FLEX1 -FLEX20	PRICE_ZONE	GREGORIANMONTH		
	ALL	GREGORIANDAY		
	FLEX1 – FLEX20			

4. For RI customers, raise a Service Request with Oracle Support when all configurations are complete, stating that the Planning Configuration processes must be executed on your environment (provide the environment name and summarize the interfaces you plan to use). Part of the configuration process directly alters the application metadata to use your specified levels, so it currently must be managed by Oracle. This step only applies if you need to use the Reporting functionality of RI. It does not apply if you are only loading data for AI Foundation applications.

Preparing Data Files

After the interfaces are configured, you must prepare the data files for upload to Retail Insights. If you are using MFP, AP, or IPO Cloud Service-Demand Forecasting Cloud Services, the data can be automatically integrated between applications. This involves a one-time setup process during your Planning implementation to enable the batch processes that move data from PDS to RI, and then enable the RI jobs to consume that data. In that case the rest of this section does not apply. Proceed to the next section on partitioning the tables instead.

If you are uploading data files directly to RI, then there are some rules and guidelines to follow when creating the files:

- All key columns on the interface must be populated, even if you have specified "ALL" as the data level. You should use a default value of -1 to populate these fields. This includes the fields PROD_DH_NUM, PROD_DH_ATTR, ORG_DH_NUM, SUPPLIER_NUM, as well as all other columns before the CAL_DATE on the flex interfaces.
- The calendar (CAL_DATE) field must always be a date. If loading the data above day level, use the end-of-period date. The format must match the date mask specified on the context (CTX) file.
- The PLANNING_TYPE_CODE field was originally used to specify if the plan type was COST or RETAIL, but this makes no functional difference in reporting at this time and can be set to any value.
- The VERSION_NUM field specifies the plan version in numerical order, starting with 0. When integrating with MFP, the Original Plan is always version 0, and the Current Plan is version 1. These two versions correlate to OP and CP metrics in Retail Insights metadata. Versions greater than 1 may be used to capture version history in the RI database, and the highest available version will be shown in the CP set of metrics.

- The DATASOURCE_NUM_ID field must be provided with a hard-coded value of 1, similar to all other RI interface specifications.
- The INTEGRATION_ID field must be provided with a unique value that identifies the record, such as a concatenation of all primary key values.
- The data file may use standard RI formatting, which uses pipes (|) as delimiters for the columns, or you may pick other delimiters as needed. You can change the file format options in the CTX file.

The accepted list of data files would align to the interfaces:

- W_RTL_PLAN1_PROD1_LC1_T1_FS.dat
- W_RTL_PLAN2_PROD2_LC2_T2_FS.dat
- W_RTL_PLAN3_PROD3_LC3_T3_FS.dat
- W_RTL_PLAN4_PROD4_LC4_T4_FS.dat
- W_RTL_PLAN5_PROD5_LC5_T5_FS.dat
- W_RTL_PLANFC_PROD1_LC1_T1_FS.dat
- W_RTL_PLANFC_PROD2_LC2_T2_FS.dat
- W_RTL_FLEXFACT1_FS.dat
- W_RTL_FLEXFACT2_FS.dat
- W_RTL_FLEXFACT3_FS.dat
- W_RTL_FLEXFACT4_FS.dat

All of the files must be packaged into a ZIP file for upload, such as RI_MFP_DATA.zip or RI_EXT_DATA.zip. The zip file for these can be marked as optional (which means the nightly batch will not wait for them to arrive before starting) or required (meaning the batch must receive them every night to start processing). It is standard to mark these files as required, since you do not want the batch to run without your planning data, or the next day's reporting could be inaccurate.

Lastly, you must also provide context (CTX) files for each interface, which specifies the exact columns you are populating with data. This allows you to provide only a subset of the many columns on the interface, as long as the key columns and at least one data column are populated. Each CTX file is named similarly to the interface (e.g. W_RTL_PLAN1_PROD1_LC1_T1_FS.dat.ctx) and contains the file format parameters such as column delimiter and header row counts, followed by a single column of values matching the interface field names. Refer to the Retail Insights Interfaces Guide for complete details on the available fields. Refer to the Retail Analytics and Planning Implementation Guide for complete details on CTX file contents.

Partition Tables

The first planning table (W_RTL_PLAN1_PROD1_LC1_T1_F) requires special partition structures due to its use in integrating data to the AI Foundation applications like Promotion and Markdown Optimization. The plan table has flexible partitions which can be either Day or Week level, depending on the data level you plan to use. You must perform this partition setup process before using the PLAN1 table, or it will not function properly during data loads. The other plan tables do not have partitioning at this time as they are only used by AIF forecasting processes which don’t need partitioned data.

In order 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 also 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 (e.g. use a value of 6M to partition 6 months into the future).

Once the configuration steps are completed, you must run the adhoc POM process CREATE_PARTITION_ADHOC (this is run once as part of any standard implementation, but if you have already run it once before updating the plan configurations, then you must run it again). Example Postman message body to send to POM via Rest API call:

```
{
"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"
}
```

Loading Plan Data

Table 3-3 Dataset Jobs

Job	Ad Hoc Processes	Usage
W_RTL_PLAN1_PROD1_LC1_T1_FS_JOB W_RTL_PLAN2_PROD2_LC2_T2_FS_JOB W_RTL_PLAN3_PROD3_LC3_T3_FS_JOB W_RTL_PLAN4_PROD4_LC4_T4_FS_JOB W_RTL_PLAN5_PROD5_LC5_T5_FS_JOB	N/A (Nightly Only)	Load the planning flat files from the server into the staging area in the database. Only used as part of nightly batch processing, flat files cannot be loaded in ad hoc processes. Tables are truncated at the beginning of each execution so new data can be inserted.
W_RTL_PLAN1_PROD1_LC1_T1_FS_SDE_JOB W_RTL_PLAN2_PROD2_LC2_T2_FS_SDE_JOB W_RTL_PLAN3_PROD3_LC3_T3_FS_SDE_JOB W_RTL_PLAN4_PROD4_LC4_T4_FS_SDE_JOB W_RTL_PLAN5_PROD5_LC5_T5_FS_SDE_JOB	LOAD_PLANNING1_DATA_ADHOC LOAD_PLANNING2_DATA_ADHOC LOAD_PLANNING3_DATA_ADHOC LOAD_PLANNING4_DATA_ADHOC LOAD_PLANNING5_DATA_ADHOC	Imports planning data directly from the MFP and AP exports from Planning Data Store (PDS) to Retail Insights. The data is inserted into the staging tables, and assumes you are not providing any data via flat file (each interface should only come from either MFP/AP or flat file but not both).

Table 3-3 (Cont.) Dataset Jobs

Job	Ad Hoc Processes	Usage
W_RTL_PLAN1_PROD1_LC1_T1_F_JOB	LOAD_PLANNING1_DATA_ADHOC	Transforms and loads planning data from staging areas into the final fact tables. Used both for the nightly processing and for ad hoc loads.
W_RTL_PLAN2_PROD2_LC2_T2_F_JOB	LOAD_PLANNING2_DATA_ADHOC	
W_RTL_PLAN3_PROD3_LC3_T3_F_JOB	LOAD_PLANNING3_DATA_ADHOC	
W_RTL_PLAN4_PROD4_LC4_T4_F_JOB	LOAD_PLANNING4_DATA_ADHOC	
W_RTL_PLAN5_PROD5_LC5_T5_F_JOB	LOAD_PLANNING5_DATA_ADHOC	
CLEANUP_C_LOAD_DATES_PLANNING_JOB	CLEANUP_C_LOAD_DATES_PLANNING_ADHOC	In order to run a planning load more than once in a single business date, you must clear the execution status from the prior run, otherwise the batch process will skip any subsequent loads (until the business date changes).

Loading Aggregate History

Special flex fact tables have also been provided to support loading of pre-aggregated history fact data for AI Foundation to use to generate forecasts for Planning. These tables should not be used unless there is no other way to provide history data except at a level above item/location. These tables will completely bypass the normal data flow for history in RI and AIF, directly populating history fact tables just for limited AIF use-cases.

These tables work identically to PLAN and FLEX tables described in the previous sections. They use FACT as the table name prefix, for example W_RTL_FACT1_PROD1_LC1_T1_F. Four such FACT tables are provided for the different data intersections that may exist for aggregate actuals. Just like PLAN tables, you must configure these in C_ODI_PARAM_VW in the Control Center using parameters like RI_FACT1_PROD_LEVEL and RI_FACT1_ORG_LEVEL.

Applying Configurations

The following information is provided as a reference for the process that Oracle follows to setup the planning and flex fact configurations. This process is man-aged by Oracle in cloud environments and would not be run by implementers directly, but it can be useful to know what changes are being made to the system when this is performed.

Create a Wallet

First create a wallet for executing the planning configuration script for WebLogic Username/Password and RPD Username/Password. RPD Username can be set to "RI_RPD" since RPD does not have a username specifically. Passing WebLogic alias and RPD alias as an argument while running script connects to WebLogic user and executes the configuration script to deploy the RPD with the RPD alias provided.

> **Note:**
>
> Reuse the wallet already created by the Installer if exists else create wallet using the steps below.

Perform the following steps to create a wallet:

1. Set the JAVA_HOME environment variable.
2. Navigate to the following directory:

   ```
   cd <STAGING_
   DIR>/ori/installer/ori/Build/orpatch/deploy/retail-public-security-api/bin
   ```

3. Add execute permission to the save_credential.sh scripts in above location.

   ```
   chmod +x save_credential.sh.
   ```

4. Create an alias for the WebLogic and RPD user by running save_credential.sh with the following arguments:

   ```
   ./save_credential.sh  -a <alias_name> -u <username> -l </location/wallet/dir>-p
   <Partiton_Key>
   ```

   Where:

   - **<alias_name>**: Alias Name for the User
   - **<username>**: Username
   - **</location/wallet/dir>**: Location where wallet needs to be created
   - **<Partiton_Key>**: RI Partition Key

   For example:

   ```
   ./save_credential.sh  -a WEBLOGIC-ALIAS -u wlsadmin -l/u01/retail/ri/wallet -p
   RI_KEY
   ./save_credential.sh  -a RPD-ALIAS -u RI_RPD -l /u01/retail/ri/wallet -p RI_KEY
   ```

RunPlanningConfigurationScript.sh

Perform the following steps to run PlanningConfigurationScript.sh:

1. Navigate to the following directory.

   ```
   cd <RETAIL_HOME>/ra_obiee_source_code/OBIEE_BAR_Deployment
   ```

2. Unzip the "OBIEE_BAR_FILE_DEP.zip" file in the above path.
3. Run the PlanningConfigurationScript.sh with the following arguments:

   ```
   ./PlanningConfigurationScript.sh RETAIL_HOME OBIEE_HOME PLANNING_CONFIG_HOME MODE
   WEB-LOGIC_USER_ALIASNAME RPD_ALIAS_NAME WALLET_PATH PARTITON_KEY EXECUTION_ MODE
   ```

   Where:

   - **RETAIL_HOME**: Retail Home path
   - **OBIEE_HOME**: OBIEE Home Path

- **PLANNING_CONFIG_HOME**: Path where all input xml's, properties.txt file, keyTableNames.txt file, All java files, jar files, PlanningConfigurationScript.sh are placed
- **MODE**: File mode (f) or Database Mode (d)
- **WEBLOGIC_USER_ALIASNAME**: Alias name of WebLogic user created in wallet
- **RPD_ALIAS_NAME**: Alias name of RPD user created in wallet
- **WALLET_PATH**: Path where wallet is created
- **PARTITON_KEY**: RI Partition Key
- **EXECUTION_MODE**: Parameter for Configuration, which we are running

Database Mode (d)

In database mode user has to first set Planning levels in C_ODI_PARAM table then run PlanningConfigurationScript.sh script with MODE argument d. Configuration levels need to be set for below list of PARAM_NAME in the C_ODI_PARAM table.

File Mode (f)

In File Mode user has to first set Planning levels in the file proper-ties.txt in the path where OBIEE_BAR_FILE_DEP.zip is un-zipped, then run PlanningConfigurationScript.sh script with MODE argument f.

> **Note:**
>
> FLEX_FACT Configuration requires setting of only FLEX_ FACT levels in properties.txt and for planning only planning levels should be set. Similarly Planning IPO Cloud Service-Demand Forecasting Forecast Integration requires setting only Planning IPO Cloud Service-Demand Forecasting Forecast Integration levels in properties.txt.

Figure 3-1 Properties.txt File

properties.txt

```
RI_FLEXFACT1_ATTR_LEVEL=ALL
RI_FLEXFACT1_CAL_LEVEL=DAY
RI_FLEXFACT1_ORG_LEVEL=LOCATION
RI_FLEXFACT1_PROD_LEVEL=ITEM
RI_FLEXFACT1_SUPP_LEVEL=ALL
RI_FLEXFACT2_ATTR_LEVEL=ALL
RI_FLEXFACT2_CAL_LEVEL=DAY
RI_FLEXFACT2_ORG_LEVEL=LOCATION
RI_FLEXFACT2_PROD_LEVEL=ITEM
RI_FLEXFACT2_SUPP_LEVEL=ALL
RI_FLEXFACT3_ATTR_LEVEL=SCENT
RI_FLEXFACT3_CAL_LEVEL=PERIOD
RI_FLEXFACT3_ORG_LEVEL=AREA
RI_FLEXFACT3_PROD_LEVEL=DIV
RI_FLEXFACT3_SUPP_LEVEL=PARENT_SUPPLIER
RI_FLEXFACT4_ATTR_LEVEL=STYLE
RI_FLEXFACT4_CAL_LEVEL=HALFYEAR
RI_FLEXFACT4_ORG_LEVEL=DISTRICT
RI_FLEXFACT4_PROD_LEVEL=GRP
RI_FLEXFACT4_SUPP_LEVEL=PARENT_SUPPLIER
RI_PLAN1_ATTR_LEVEL=C
RI_PLAN1_CAL_LEVEL=YEAR
RI_PLAN1_ORG_LEVEL=REGION
RI_PLAN1_PROD_LEVEL=SBC
RI_PLAN1_SUPP_LEVEL=SUPPLIER
RI_PLAN2_ATTR_LEVEL=B
RI_PLAN2_CAL_LEVEL=WEEK
```

> **Note:**
>
> While running in filemode, planning levels to be set in properties.txt should match the C_ODI_PARAM table planning level entries.

The following values are valid for the properties.txt file:

Organization:

- COMPANY
- CHAIN
- AREA
- REGION
- DISTRICT

- LOCATION
- CHANNEL
- PRICE ZONE
- ALL

Product:

- CMP
- DIV
- GRP
- DEPT
- CLS
- SBC
- ITEM
- ITEM LEVEL1
- ALL

Calendar:

- YEAR
- HALFYEAR
- QUARTER
- PERIOD
- WEEK
- DAY

Supplier:

- SUPPLIER
- PARENT_SUPPLIER
- ALL

Product Attributes:

- STYLE
- FLAVOR
- SCENT
- FABRIC
- SIZE
- C (C denotes Color)
- B (B denotes Brand)

PlanningConfigurationScript.shScript

The PlanningConfigurationScript.sh script executes the following three operations:

- Downloads the deployed RPD with name default.rpd
- Executes jar file to modify RPD reading planning levels from properties.txt (file mode) or C_ODI_PARAM table (Database Mode).
- Uploads modified RPD (default.rpd).

4

Xstore Sales Integration

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

Sales from XStore are currently displayed in reporting for the current Business Date+1. For example, if your audited sales have been loaded up to the business date of 09/21/2020, then the Xstore sales will be displayed in reporting only for 09/22/2020. Prior days of Xstore sales will not be shown, because it is assumed your audited sales will include all prior transactions. Prior days of Xstore sales will be kept in the database for a limited amount of time before being purged. It is also important to note that this integration is specifically for sales only, RI does not process other transaction types such as timeclock, pre-order, work order, till, open/close, and other non-sales types (except when one of those transactions results in a completed sale where customer payment is collected).

Data Load Process

The data first comes from Xstore to the Retail AI Foundation data model using a web service API. The API is configured as part of AI Foundation but is leveraged by Retail Insights to get the raw XML POSLOGs into the database for transformation to our data model. The steps below highlight the flow of data into RI.

1. The XML is first extracted from the AI Foundation's staging area for the API into the RI database table W_RTL_POSLOG_XML_G.
2. Next, the data for the current load process is moved to temporary tables W_RTL_SLS_POS_IT_LC_DY_TMP and W_RTL_SLS_POS_TAX_TMP. The current set of records is defined as all unprocessed XMLs that have been received since the last time the intraday batch process was run. For example, if the process runs hourly then for each iteration of this cycle, one hour of transactions will be processed in a single set.
3. The XML data is transformed from raw XML into the RI relational data model, filtered to the subset of sales transaction types we support, and then inserted to W_RTL_SLS_POS_IT_LC_DY_FS.
4. Lastly the data is loaded from W_RTL_SLS_POS_IT_LC_DY_FS to W_RTL_SLS_POS_IT_LC_DY_F. This load process maps all of the primary key values in the transaction record to the RI dimensions and creates the links across all RI tables that are necessary to report on the data in OAS.
5. Transactions which don't have matching keys on one or more of the RI dimensional tables (for example, a product that doesn't exist) will be rejected by the final load step and will not appear in the reporting layer. This will be the case for the core dimensions of product, location, and calendar. Employees are also matched against RI data but they have the option to be auto-seeded into our data model so that we do not reject records for new/unknown employee IDs.

The program used to perform this intraday processing is LOAD_POSLOG_DATA_ADHOC, which can be found in the Ad Hoc area of the POM batch schedule for Retail Insights. Using the ad hoc scheduling capabilities of POM, you would configure this job to run on a set frequency throughout the day. The actual frequency will largely depend on the trade-off of performance and data volumes. For example, if it takes X minutes to process a block of transactions which come in over Y minutes, X cannot be greater than Y or RI would not have enough time to complete a load before starting the next one. Identifying the optimal run frequency will need to be worked out during the implementation using realistic daily volumes.

Supported Data Types

Only certain types of Xstore transactions are processed into RI as sales. All other transaction types are ignored by the data load process in order to provide equivalent comparisons of audited and unaudited sales data between our two subject areas. The general logic for this comparison is that a sale is recorded to the data warehouse when customer payment is collected and the item ownership has been transferred. A sale should show an equivalent reduction in inventory and both the sale and inventory changes would be recorded on the stock ledger at the end of the day. The only exception to this would be sales of services, which should be tracked as non-inventoried or non-merchandise items so that RI is still aware of the dimensions present on the transaction before it occurs. Review the transaction usage details below.

Table 4-1 Transaction Usage Details

Transaction Type	Usage Details
Sale	Processed as normal sale into RI SLS_* columns in the database.
Return	Processed as normal sale into RI RET_* columns in the database. Net sales in RI is calculated by subtracting return values from sales values, so both sales and returns are held in the database as positive values.
Exchange	An equal exchange transaction will have it's lines condensed to show values both on SLS_* and RET_* columns in the database for that item.
Special Order (Complete)	A completed special order will be captured as a sale based on the PreviousCustomerOrder status.
Work Order (Pickup)	A completed work order will be captured as a sale based on the PreviousCustomerOrder status.
Hold Account (Pickup)	A pickup on a hold account will be captured as a sale based on the PreviousCustomerOrder status.
Pre-Sale (Pickup)	A pickup on a pre-sale will be captured as a sale based on the PreviousCustomerOrder status.
Send Sale (Complete)	A completed send sale will be captured as a sale based on the SaleForDelivery status.
Layaway (Complete)	A completed layaway will be captured as a sale based on the PreviousLayaway status.

Supported Dimensions

Only certain dimensions in RI are supported when reporting on Xstore sales data, based on what we expect to have in our daily interfaces with other applications such as RMFCS. Review the list of supported dimensions below.

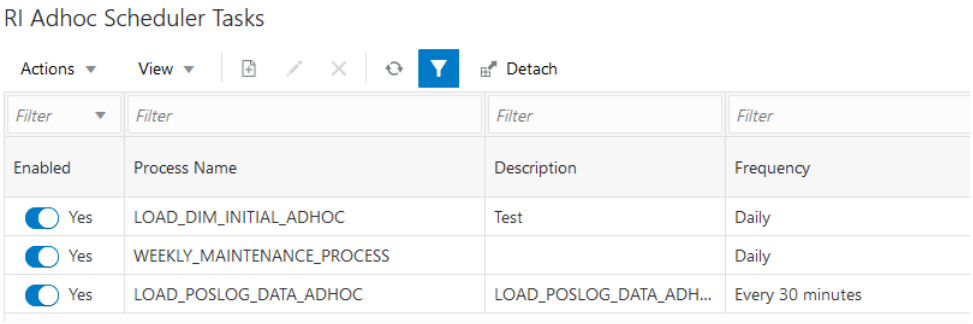
Table 4-2 Supported Dimensions

Dimension	Usage Details
Item	The item ID listed on a Line Item of the transaction must be a valid item in the RI Item hierarchy in order for us to load the record.
Organization	The location ID where the transaction was captured must be a valid location in the RI Organization hierarchy in order for us to load the record.
Business Calendar Gregorian Calendar	The date when the transaction occurred will be captured and reported using the Fiscal Calendar or Gregorian Calendar. POS sales are only shown for the current business date + 1.
Time of Day	The hour and minute when the transaction occurred will be captured and reported using the Time of Day dimension. The POS Sales fact itself also as a supplemental Timestamp attribute for more granular analysis if needed.
Supplier	The primary supplier associated with an item can be used to report on and aggregate the POS sales data. The supplier-item relationship is a separate feed into RI and not part of the POS data.
Retail Type	The retail type of a POS transaction will depend on the contents of the transaction as well as the other data in RI. By default, a transaction will be categorized as a Regular (R) sale. If the transaction has a Price Modifier of type 'Promotion' then it will become a Promotional (P) transaction. If you are providing inventory data with clearance flags to RI, then we can leverage that additional detail to flag sales as Clearance (C) sales.
Employee	RI reports on employees using either the Cashier or Salesperson attributes, and both may be used with POS sales data assuming the cashier/salesperson is identified on the transaction. Because employee data is not maintained by a merchandising system, RI allows these records to be created automatically if we encounter an unknown employee ID on a transaction.

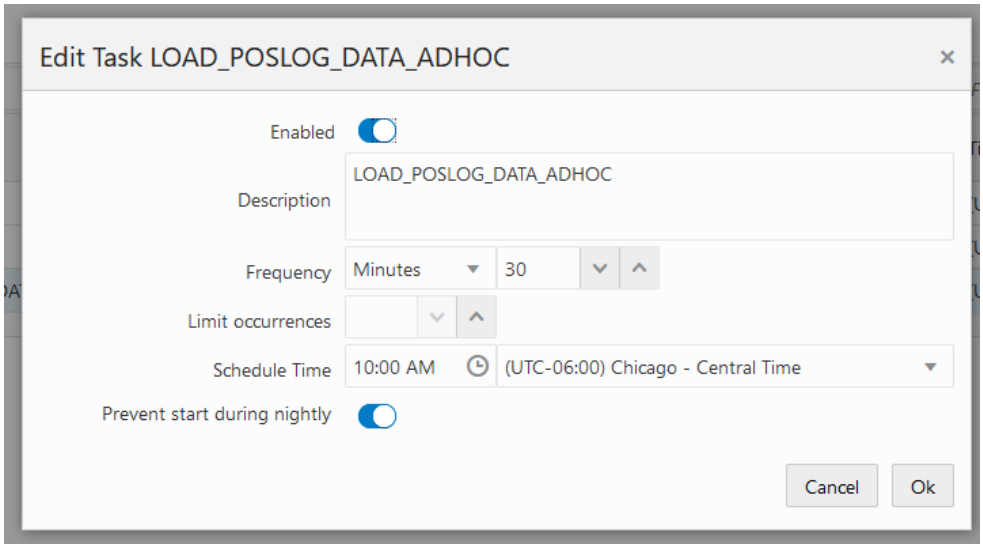
Intraday Scheduling

Due to the real-time nature of POSLOG data from Xstore, an intraday process must be used to move the data from it's raw XLML format into the Retail Insights relational data model throughout the day. This intraday process can be maintained in the POM application alongside the RI nightly batch. Follow the steps below to enable this process.

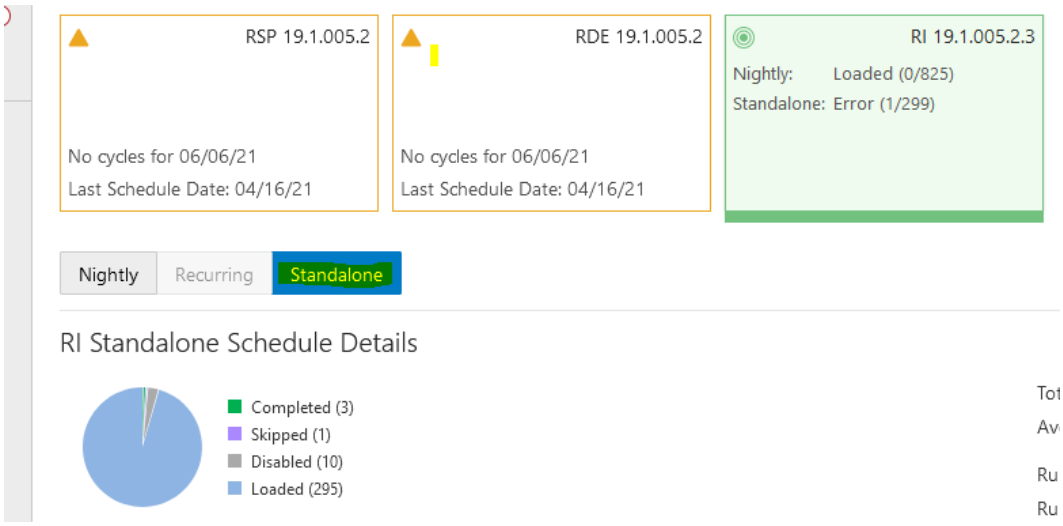
1. Login to the POM application UI as a user with batch scheduling permissions.
2. Click on the Scheduler Administration link under the Tasks menu.
3. Click on the RI schedule. Make sure the LOAD_POSLOG_DATA_ADHOC process is present in the Standalone tab and enabled.



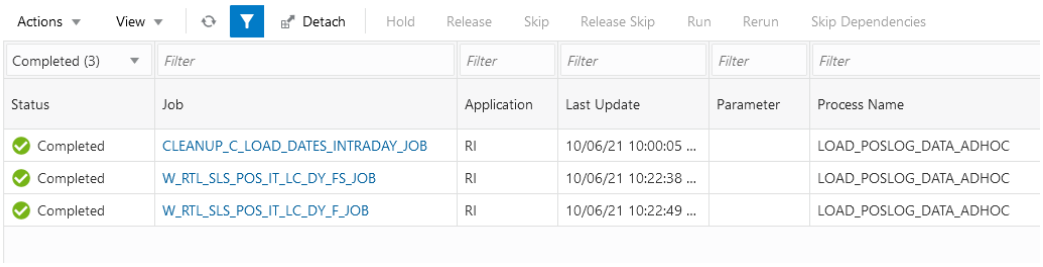
4. Highlight that row and click the Edit action icon.



5. Adjust the frequency to the desired interval, such as 30 minutes or 1 hour. Click OK to confirm the change.
6. Navigate to the Batch Monitoring screen and Restart Schedule for the RI Standalone schedule.



7. Once restarted, the adhoc process will begin to run automatically on the specified interval.



5
Internationalization

Internationalization is the process of creating software that is able to be translated more easily. Changes to the code are not specific to any particular market. Retail Insights has been internationalized to support multiple languages.

> **Note:**
>
> Retail Insights uses DB language code and not ISO codes for all the supported languages. Retail Insights will look up language codes from RDE. If, in the case a language supported by Retail Insights is not available in the source system, then the language under SRC_LANGUAGE_CODE will be used as the local language.

This section describes configuration settings and features of the software that ensure that the base application can handle multiple languages.

Translation

Translation is the process of interpreting and adapting text from one language into another. Although the code itself is not translated, components of the application that are translated may include the following:

- Graphical user interface (GUI)
- Error messages
- Application metadata (metric names and descriptions)

The following components are not translated:

- Documentation (online help, release notes, installation guide, user guide, operations guide)
- Batch programs and messages
- Log files
- Configuration tools
- Demonstration data and sample reports
- Training materials

The user interface for Retail Insights is provided by Oracle Analytics (OAS). The current list of languages for OAS can be found here:

https://docs.oracle.com/en/cloud/paas/analytics-cloud/acabi/switch-different-language.html

The metadata objects (RPD) for Retail Insights has been translated into:

- Arabic
- French

- German
- Italian
- Polish
- Portuguese (Brazilian)
- Spanish

Multi-Language Setup

Retail Insights data is supported in 18 languages. This section provides details of various scenarios that may come across during implementation. See "Translation" for a list of supported languages.

Since multi-language data support in Retail Insights is dependent on the availability of the multi-language data in the source system, it is important to understand various scenarios the user may encounter. Before proceeding review the following facts about multi-language support:

- Retail Insights programs extracts multi-language data from source systems.
- A list of languages for multi-language data support can be chosen during the installation process. Please refer to the *Oracle Retail Insights Installation Guide* for more details.
- Depending on the implementation, the source system may or may not have data for particular supported language(s). For example, RMS supports Item Descriptions in multiple languages but the item's description may not be available in the translated languages.
- For source system released languages, please refer to source system Operations Guides.
- You must select a Retail Insights primary language for data purposes to be supported within the source system.
- You must delete not needed languages from W_LANGUAGES_G in Retail Insights, to improve batch performances.

Refer to the *Oracle Retail Home Administration Guide* for information on how to setup and use resource bundles for custom strings modifications.

Scenario 1

All the supported languages are implemented in Retail Insights and the same set of languages are supported in the source system as well.

Multi-lingual data sets are enabled in both Retail Insights and the source system.

Data Scenario 1a

Translated data exists for all records in Source System: This is an ideal scenario where the source system supports data for the same set of languages as Retail Insights and data for the required column exists in all the languages in the source system.

In this scenario the attributes that are supported for multi-languages will get all the multi-language data in Retail Insights.

Data Scenario 1b

Translated data does not exist for some of the records in the source system.

For the attributes for which data is not available in the source system, Retail Insights will display the attribute in source system primary language. For example, Retail Insights requests data in German and English languages. In RMS the Item attribute description is not available in the German language but is available in English language.

Retail Insights will display Item description in English to a user who is logged into Oracle Analytics (assuming English is the primary language of RMS for that implementation).

Scenario 2

All or a subset of languages are implemented in Retail Insights and some of these are not supported in the source system:

Data Scenario 2a

Translated data does not exist for some of the languages in the source system. In this case, the data is displayed in Retail Insights' primary language.

Scenario 3

Source system supports more languages than are supported for Retail Insights. In this case Retail Insights filters out the additional languages' data. This data will not be loaded into Retail Insights tables and cannot be used for reporting.

6
Compression and Partitioning

This chapter describes how Retail Insights implements compression and offers a discussion of Oracle partitioning.

Overview of Compression

Although data warehouses are often very large, the amount of detail generated in some Retail Insights tables is enormous even by usual standards. That is, a retailer with 500,000 items and 500 locations would generate 250,000,000 new rows each day. Storing this amount of uncompressed data is impractical from a disk storage perspective, in the cost to store the rows, the cost to perform backups, and other database maintenance operations.

One approach that Retail Insights uses to reduce the data volume is compression. This chapter describes:

- What compression does
- Mechanics of compression
- Which tables are currently compressed
- Oracle features that are related to compression
- Strategies for implementing compressed tables

What Compression Does

Compression refers to storing physical data that only reflects changes to the underlying data source, and filling in the gaps between actual data records through the use of database views. This method is engaged primarily for subject areas that are perpetual, such as inventory. That is, when querying sales data, a valid sale record exists (a sale occurred) or a record does not exist (no sale occurred). However, when querying for on-hand inventory, even if no change occurred to the inventory on the date desired, a valid value is still required. One way to resolve this discrepancy is to store a record for every day and a valid item-location combination as mentioned above. Another method, compression, allows for the storage of only changes to the inventory position. The query is resolved by looking backward through time from the desired date (if no change record exists on that date) until an actual change record is found. This method returns the correct current data with the minimum requirements necessary for processing and storing data.

Retail Insights compression is different with Oracle DB table compression. Oracle DB table compression compress data by eliminating duplicate values within a data block. Any repetitive occurrence of a value in a block is replaced by a symbol entry in a "symbol table" within the data block. So for example DEPT_NUM=10 is repeated five times within a data block, it will be only stored once and for the other four times a symbol entry will be stored in symbol table. Oracle database table compression can also significantly reduce disk and buffer cache requirements for database tables while improving query performance. Oracle database compressed tables use fewer data blocks on disk, reducing disk space requirement.

Mechanics of Compression

The purpose of decompression views is to give the application the illusion that there is a record for each possible combination (that is, an item-location-day record for each permutation), when in fact there is not. Thus, the fact of whether a table is compressed or not should not be visible to the application that queries data from that table. A compressed table is made up of two distinct parts: a 'seed' that consists of all existing combinations at a point in time (typically the first day or week of the table or partition) and the changed data since that time. Retail Insights compressed tables use FROM_DT_WID and TO_DT_WID columns to indicate the time range in which records are valid. When resolving a query for a particular record, the decompression view provides the latest record for the requested item and location with the maximum day that is less than or equal to the requested day. A decompression view needs to encompass both the seed and all of the changed data since that seed. A decompression view compares FROM_DT_WID and TO_DT_WID of records with FROM_VALUE and TO_VALUE on partition mapping table W_RTL_PARTITION_MAP_G to make sure that a right partition is used by the decompression view.

To illustrate how the decompression views actually work, assume the following:

- The user is interested in the inventory position of item 10 at location 10 on 1/23/02.
- The seed was done on 1/1/02. Changes were posted on 1/4/02, 1/15/02, and 1/30/02.
- The row that is presented to the application by the decompression view is the row on 1/15/02, because it is the latest date that is less than or equal to the requested date.

As a second example, assume that the inventory position of item 10, location 10, day 1/3/02 was desired. Because there was no change record less than or equal to the desired date, the seed record from 1/1/02 will be presented to the application.

Compression's performance is excellent when the user is querying for a single day (as in the example above). When querying over a group of days, however (that is, all of the inventory positions at a given location on a given day), the performance can be unacceptable. Even though the user is requesting a group of information back, and in most cases the database can process groups of information efficiently, each individual row must be evaluated individually by the decompression view and cannot be processed as a group. To counteract the slow performance of these summary operations, you may take advantage of compressed table partition seeding (see "Overview of Partitioning Strategies").This partition seeding utilizes the latest position status tables (also known as 'current' tables). An example is the W_RTL_INV_IT_LC_G table, which holds the current decompressed position for every item and location on the W_RTL_INV_IT_LC_DY_F table. This position can be used as a partition seed. This position is also utilized by base Retail Insights code during major change fact seeding.

Compressed Tables and 'CURRENT' Tables

The table below illustrates the compressed tables within Retail Insights, along with their corresponding 'CURRENT' tables.

Table 6-1 Compressed Tables and CURRENT Tables

Compressed Tables	Current Tables
W_RTL_PRICE_IT_LC_DY_F	W_RTL_PRICE_IT_LC_G
W_RTL_BCOST_IT_LC_DY_F	W_RTL_BCOST_IT_LC_G
W_RTL_NCOST_IT_LC_DY_F	W_RTL_NCOST_IT_LC_G
W_RTL_CO_LINE_STATUS_F	W_RTL_CO_LINE_STATUS_G
W_RTL_COMP_PRICE_IT_LC_DY_F	W_RTL_COMP_PRICE_IT_LC_G
W_RTL_INVU_IT_LC_DY_F	W_RTL_INVU_IT_LC_G
W_RTL_INV_IT_LC_DY_F	W_RTL_INV_IT_LC_G
W_RTL_PO_ONALC_IT_LC_DY_F	W_RTL_PO_ONALC_IT_LC_G
W_RTL_PO_ONORD_IT_LC_DY_F	W_RTL_PO_ONORD_IT_LC_G

Coping with Slowly Changing Dimension Type 2

Fact Close Program (factcloseplp.ksh)

On a compressed fact table, a record is only posted to the table when there is a change in one of the fact attributes. If there is no activity, no record is posted. Decompression views then fill in the gaps between physically posted records to ensure that a fact record appears for each item-location-day combination in the user interface. However, when an item, location, or department is closed or major-changed, any fact record with those dimensions becomes inactive. The decompression views need to be informed to stop filling in the gap after the last record was posted. To accomplish this instruction, scenario PLP_RetailFactCloseFact (called by factcloseplp.ksh) first queries the W_RTL_PROD_RECLASS_TMP and W_RTL_ORG_RECLASS_TMP tables to determine the compressed item-location facts that need to be closed today. The PLP_RetailFactCloseFact scenario then updates TO_DT_WID to the current date WID to stop the record. The decompression view fills in records up to the day that is in the range between FROM_DT_WID and TO_DT_WID.

Fact Open Program (factopenplp.ksh)

Retail Insights Data Compression tables require seeding when a major change in the product and organization dimension causes new surrogate keys to be created for items or locations. Seeding the compressed tables is required because the new key represents a new hierarchy relationship. If the new key is not represented on the compressed table, the compression view does not pick up any data from the day the old dimensions were closed to the day a record with the new dimensions is posted to the compressed fact tables. This missed data causes inaccuracy in query results and incorrect data aggregation.

To accomplish this seeding scenario, PLP_RetailFactOpenFact (called by factopenplp.ksh) first queries the W_RTL_PROD_RECLASS_TMP and W_RTL_ORG_RECLASS_TMP tables to determine what compressed item-location facts need to be closed today. The PLP_RetailFactOpenFact scenario then inserts seeded (closed) records for tomorrow's FROM_DT_WID, indicating that the closed fact records are no longer valid beginning tomorrow, when the newly seeded records (from PLP_RetailFactOpenFact) become active. In the case of the compressed week table, W_RTL_INV_IT_LC_WK_A, PLP_RetailFactOpenFact inserts seeded records with next week's warehouse ID.

Oracle Table Compression

Oracle table compression not only helps customers save disk space, it also helps to increase cache efficiency since more blocks can fit in the memory. Advanced Compression is available for Oracle 11g Enterprise Edition and Hybrid Columnar Compression is available for Exadata only.

Since compression could cause contention when tables get updated, it is suggested users only compress non-current partitions and leave the current partition uncompressed. This partial compression approach has proven to be a valuable implementation option.

Overview of Partitioning Strategies

This section describes partitioning strategies for Retail Insights data marts. Although optional, partitioning provides powerful performance benefits, and therefore is highly recommended. Tables in the RA_partitioned_tables.xls spreadsheet (see the *Oracle Retail Insights Installation Guide*) are highly recommended to be partitioned. If a report runs slowly and a fact table in the query is not partitioned, that fact table may be a good candidate for partitioning. For large tables, such as the inventory, pricing, cost and sales tables, splitting them into table partitions can provide the following benefits:

- Partitions are smaller and therefore easier to manage.
- Management operations on multiple partitions can occur in parallel.
- Partition maintenance operations (such as index rebuilds) are faster than full table operations.
- Partition availability is higher than table availability (that is, when recovering a particular partition, users may access all other partitions of the table at the same time).
- The optimizer can prune queries to access data in only the partition of interest, not the entire table (that is, if you are interested only in February's data, you do not need to look at any of the table's data outside of the February partition).
- Partitions are separate database objects, and can be managed accordingly (that is, if December sales are frequently accessed throughout the year whereas other months are not, the December sales partition could be located in a special tablespace that allows for faster disk access).
- In some situations, the Oracle database can create parallel operations on partitions that it cannot on tables; an example is joining between two different tables if they are partitioned on the same key (this feature is called a 'parallel partition-wise join').

Indexes, as well as tables, can be partitioned. Index partitions can be global (one index over the table, regardless of whether the table is partitioned or not) or local (there is a one-to-one correspondence between index partitions and table partitions). In general, when tables are partitioned, local indexes should be preferred to global indexes for the following reasons:

- Maintenance operations involve only one index partition instead of the entire index (that is, if the oldest table partition is aged out, a local index partition can be dropped along with its corresponding table partition, whereas an entire global

index will need to be rebuilt after it becomes unusable when a table partition is dropped).

- The optimizer can generate better query access plans that use only an individual partition.
- When multiple index partitions are accessed, the optimizer may choose to use multiple parallel processes rather than just one.

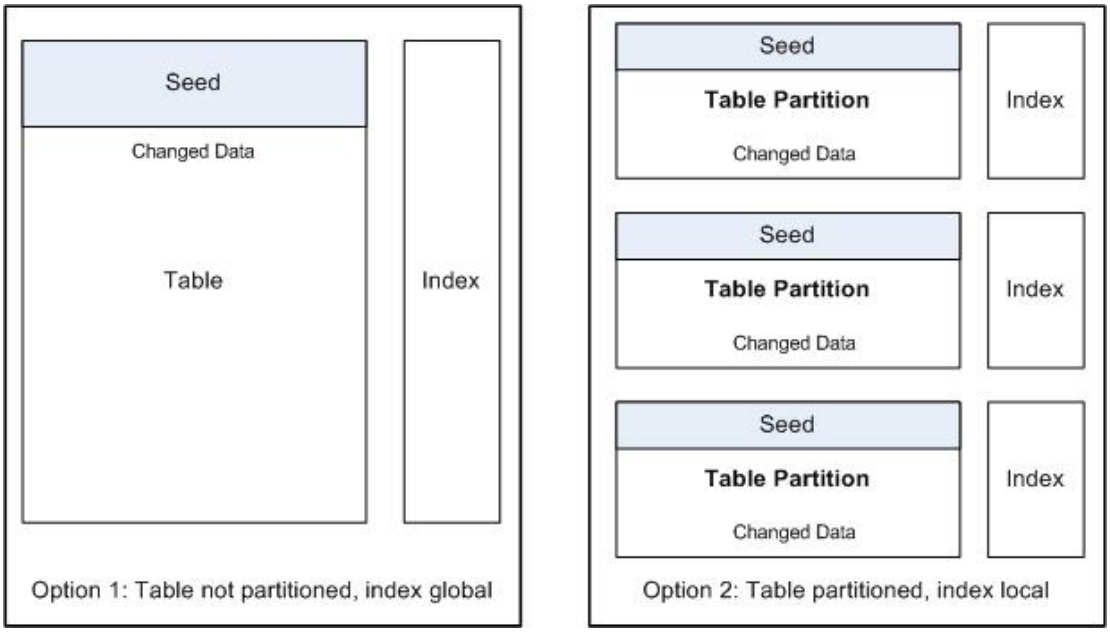
Implementing Retail Insights Partitioning

For retailers who choose to partition a fact table, the figure on the following page illustrate some of the possibilities for table and index layout.

In general, option 2 is the preferred solution for large regular or compressed tables (for example, the W_RTL_INV_IT_LC_DY_F and W_RTL_PRICE_IT_LC_DY_F tables). It uses table partitions and local indexes, thus minimizing the impact of index maintenance and the deletion of old table partitions. Global indexes on partitioned tables are not recommended.

Option 1 can be used for smaller compressed tables. The disadvantage is that, functionally, there is no way to delete historical data and the table continues to grow.

Figure 6-1 Retail Insights Partitioning Options



Setup and Maintenance for Partitioning Retail Insights Compressed Inventory Table

The following procedure describes how to setup and maintain Retail Insights partitioning of the compressed inventory table (W_RTL_INV_IT_LC_DY_F) using Retail Insights partitioning. Note that these steps are informational and are managed by Oracle in cloud environments.

1. Make the following determinations, among others (see the *Oracle Retail Insights Installation Guide* for details):

- Your partitioning strategy.
- The time period your partitions will use.
- The 'values less than' boundaries according to your multi business calendar WID values.
- How many partitions are to be used.
- The partition naming standard.

2. On the database, create the partitions and indexes for the tables you want to partition.
3. Verify you have populated the Time Calendar Dimension. See the *Oracle Retail Insights Installation Guide* for details.
4. Perform step numbers 2 and 3 whenever any of the following events occur:
   - Records are added to or deleted from the Time Calendar tables W_MCAL_DAY_D (extending time calendar for a new time period).
   - Partitions are added to the Inventory Position table W_RTL_INV_IT_LC_DY_F.

Other maintenance activities include archiving and removing of partitions.

Implementing Partitioning for Compressed Inventory Table

Once the tables (including partitions) and indexes have been created, the data must be loaded. For tables that have a corresponding current status table (such as W_RTL_INV_IT_LC_DY_F and W_RTL_INV_IT_LC_G), the following steps are recommended:

> **Note:**
>
> All these steps can be performed automatically by the Retail Insights seeding program PLP_RetailPartSeed.ksh. See the *Oracle Retail Insights Operations Guide* for detail about how to execute this script.

1. In the partition mapping table W_RTL_PARTITION_MAP_G, update column TO_VALUE with the current business date WID for the latest partition on the target table W_RTL_INV_IT_LC_DY_F.
2. Insert a new record to partition mapping table W_RTL_PARTITION_MAP_G with next business date WID or week WID as FROM_VALUE and dummy value '999999999999999' as TO_VALUE. Column TABLE_NAME must be populated with target table W_RTL_INV_IT_LC_DY_F and column PARTITION_NAME must be populated with 'P_XX'.

> **Note:**
>
> XX is the number part of current partition name on the same target table plus 1. This partition name 'P_XX' can be different from the real partition name used in the database.

3. Copy data in W_RTL_INV_IT_LC_G table as the seed to the first partition or a new partition that is going to be used on the next day.

   At this point, only the changed records are added to the W_RTL_INV_IT_LC_DY_F table. Whereas the W_RTL_INV_IT_LC_G table is a full and uncompressed version that holds the current inventory position as of the last time period.

4. When a partition boundary is crossed, the W_RTL_INV_IT_LC_G table is copied as the seed to the new partition, via the PLP_RetailPartSeed.ksh program.

If you have questions about how to implement partitioning with compression or require assistance implementing partitioning, contact Oracle Customer Support or Oracle Retail Services.

Partitioning Automation Implementation

For newly created aggregate and planning tables, we run run_partition_util.ksh once before the batch cycle to create partitions as an initialization step during implementation.

Three new tables were created to manage this process, as shown below.

- C_MODULE_ARCHIVE_TBSP

  This is for future. Currently, archiving of tablespaces are disabled.

- C_MODULE_ARTIFACT

  This tables lists the modules that are supported by RI and those that are partitioned already.

- C_MODULE_EXACT

  This tables holds the explicit search criteria for the tables that belongs to specific modules. By default: RI uses "Sales Promotion" (SLSPR), Supplier (SUPP) and Customer Order (CO) using defined search logic.

1. In the C_MODULE_ARTIFACT table, enable modules as applicable. By default, all modules will be set to 'N'. Eg: if sales and inventory are available, then set the ACTIVE_FLG = 'Y' for these modules.

   a. Set SCHEDULE_EOM_DAYS (Default Set to 0)

      Valid values are 0-6 with 0 = Saturday, 1 = Sunday,….6=Friday.

   b. Set SCHEDULE_EOW_DAYS (Default set to (null)).

   c. Enable ACTIVE_FLG for the modules where it is applicable. For example, if sales and inventory are available, then set the ACTIVE_FLG = 'Y' for these modules. By default all modules are set to 'N'.

2. Set the RI_PART_DDL_CNT_LIMIT parameter. This parameter is set in C_ODI_PARAM. By default it is set to 10000. This limits the number of statements that can be executed. This is configurable.

3. Perform the initial run to extend partitions for the newly created tables before the batch cycle.

   run_partition_util.ksh – This is the shell script that runs the partition program. It needs to be run to ensure that the required partitions are created before the batch cycle.

4. Set the RI_PART_COMPRESSION_PERIOD parameter. This parameter is set in C_ODI_PARAM. By default it is set to 7M. This compresses records which are older than 7 Months (Periods). This is configurable.

5. Set the RI_PART_RETENTION_PERIOD parameter. This parameter is set in C_ODI_PARAM. By default it is set to 37M. This purges records older than 37 Months(Periods). This is configurable.

Partitioning Automation Operation

run_partition_util.ksh – This is the shell script that runs the partition program. This programs is scheduled to run after the batch cycle email notification is sent out.

How Oracle Implements Partitions

This section highlights how partitions are implemented in an Oracle data warehouse.

For details on partitioning concepts, refer to the chapter Partitioning in Data Warehouses in the *Oracle Database Data Warehousing Guide 11g Release 1 (11.1)*.

Range partitions in the Oracle data warehouse/database are split by a range of values on the partition key. Examples include partitions by month, partitions by department number, and partitions by item range. Partitioning options also include hash partitions (spreading the rows across a fixed number of partitions by applying a hash function to the partition key), and composite partitioning (a combination of range partitioning and hash partitioning). It is recommended that you partition the tables using range partitioning. Oracle Retail also recommends that the partition key be the date field in the primary key to allow partitions to be aged out when no longer needed.

As a general guideline, partitioning must be considered for tables listed in the RA_partitioned_tables.xls spreadsheet (see the *Oracle Retail Insights Installation Guide*) and any fact tables in a slow-running query. There is an administrative trade-off between having more partitions to manage and obtaining the benefits of partitioning.

The actual physical layout of partitions varies from site to site. A general approach is to put each partition into its own tablespace and map each tablespace to a separate mount point. This has several advantages:

- Maintenance operations, as well as tablespace recovery, can occur on a partition while other partitions are unaffected.
- If manual performance tuning of the data files is being done, tablespaces and their files can be moved around to achieve optimal performance.
- If partitions are no longer being updated, their tablespaces can be changed to READ ONLY, which significantly reduces backup requirements.
- Separate mount points pointing to a separate set of physical drives significantly reduces I/O time.

Partitions are ordered from low values to high values. The partition key value for a partition is a non-inclusive upper bound (high value) for that partition. That is, if the W_RTL_SLS_IT_LC_DY_A table is partitioned by month, the high value for January, 2010, partition is 01-Feb-2010. A low value can always be inserted into the lowest partition. However, you may not be able to insert a high value depending on the high value of the highest partition. For instance, if the highest partition has a high value of 01-Feb-2000, and you attempt to insert a record with a date of 01-Feb-2010, the row will not be inserted into the table (the high value of 01-Feb-2010 is a non-inclusive upper bound). For this reason, a special high value partition with a key of MAXVALUE is available in the Oracle database. It is recommended that all partitioned tables include a dummy partition with a MAXVALUE high value.

There are special considerations for the partitioning of Retail Insights compressed tables. The following is a brief description of the different partition maintenance commands. Refer to the current Oracle database documentation set for more details:

- ADD PARTITION: Adds a new partition to the high end of a partitioned table. Because it is recommended to have a MAXVALUE partition, and this is the highest partition, the ADD PARTITION functionality can be achieved by performing a SPLIT of the MAXVALUE partition instead.
- DROP PARTITION: Drops the partition. This is the typical method to delete the oldest partitions (those with the lowest values) as they age to maintain a rolling window of data.
- EXCHANGE PARTITION: Converts a non-partitioned table into a partitioned table or converts a partitioned table into a non-partitioned table.
- MERGE PARTITION: Merges two adjacent partitions into one.
- MOVE PARTITION: Moves a partition to another segment; this is used to defragment a partition or to change its storage characteristics.
- SPLIT PARTITION: Splits an existing partition by adding a new partition at its low end.
- TRUNCATE PARTITION: Removes all rows from the partition.

Oracle database automatically maintains local index partitions in a 1-to-1 correspondence with their underlying table partitions. Any table partition operations, such as ADD PARTITION, also affect the relevant index partitions.

Summary

Partitions are useful for breaking up large tables into smaller, more manageable pieces. Take note of the following partitioning recommendations when using Retail Insights in an on-premise implementation:

- Consider partitioning tables that are in the RA_partitioned_tables.xls spreadsheet (see the *Oracle Retail Insights Installation Guide*) and fact tables that are in a slow- running query.
- Use the date as the partition key for range partitioning.
- When tables are partitioned, make their indexes local.
- Consider putting each partition in its own tablespace and each tablespace on its own mount point.
- After updates on a partition cease, consider changing its tablespace to READ ONLY to reduce backup requirements.
- If partitioning compressed tables, be sure to address any special requirements for seeding.

7
Performance

Retail Insights is a high performance data warehouse, capable of moving and storing massive amounts of data, and providing efficient access to that data via the delivered and custom built reports. For any BI solution, including Retail Insights, smart decisions on how to implement and run your data warehouse will ensure that you are getting the most out of it. This chapter contains information that will help you get the best performance out of Retail Insights and identifies common contributors that can weaken performance, as well as best practices that will ensure Retail Insights is running in the most optimal manner.All implementations are unique and the factors that are beneficial for one implementation may not have the same effect for all the implementations. It is a good practice to test several settings/approaches for the factors and recommendations listed below and use the ones that work best for your environment. The factors listed in this chapter are the key factors that impact performance but no absolute values or settings can be provided for implementation purposes due to the uniqueness of each environment.Oracle Retail Insights includes ODI for extract, transform and load and Oracle Analytics (OAS) for analytic reporting purposes. The recommendations in this chapter will focus on both back end (ETL) and front end (Oracle Analytics) components of Retail Insights. In the cloud solution, these features will be handled automatically by Oracle.

Key Factors in Performance

Based on the complexity of the report, Oracle Analytics sometimes generates complex SQL, causing the Oracle Database to pick a less than optimized execution plan. In order to avoid this scenario, it is recommended that the "SQL Plan Baseline" functionality of the Oracle 12c be enabled (it is disabled by default). For more details refer to the *Oracle 12c Performance Tuning Guide*.

Purging and Archiving Strategy

With an increased use of the Retail Insights application, the data volumes will grow and may result in slower performance. The performance impact can be on Retail Insights batch that loads data to data warehouse tables, Retail Insights reports, and storage.Adoption of purging and archiving strategy help in reducing data volumes, resulting in better performance. Consider the following recommendations while implementing these strategies in a data warehouse:

- Design your archiving and purging strategy as early as possible in the Retail Insights implementation. This helps in designing the most optimal table partitioning for large tables.
- Ensure that the data is deleted in the most optimal manner. SQL delete statements may not be the most efficient way of removing unnecessary data from Retail Insights tables. Consult with your database administrator to discuss purging and archiving techniques.
- Purging and archiving of tables must be carefully designed as it requires a good understanding of analytic reports required by business users or regulatory requirements that require companies to retain certain data for a required duration. For example, in certain cases, aggregated data may be kept longer as compared to the base level fact

data because the users are interested in summary level reports as compared to detailed (base level) reports for data older than two years.

- Automation of the archiving and purging processes ensures that a consistent approach is being followed in maintaining tables with large data volumes and provides consistent report performance to the users.
- While designing purging programs, make sure that dimensional data is not deleted for which fact data is available or will be available.
- An important consideration during purging is to make sure that Retail Insights seed data (where applicable) is not deleted accidentally.

Flexible Aggregates

Retail Insights, by default, provides several aggregate tables. For the complete list, see "Aggregates List". These pre-built aggregate tables are selected based on the following:

- General usage patterns of the data
- Reporting needs (As-Is or As-Was or both)
- General aggregation ratio

The ratio between data in the base fact table versus data in the potential aggregate table should be considered while deciding whether the fact table should be aggregated or not. A ratio of 1:5 through 1:10 is a good starting point, a ratio of 1:10 through 1:20 is good to aggregate, and a ratio of 1 to more than 20 must be aggregated.

During implementation or before, it is expected for the retailer to identify these scenarios and select the appropriate aggregate tables for best performance and usability. All the aggregate tables which are pre-packaged will have the ODI and Oracle Analytics mappings. It is highly recommended not to use Retail Insights with all the available aggregates.

Using all these aggregations improves report performance but the improved report performance should be weighed against reduced ETL batch performance and increased storage requirement.

The reason for providing these aggregates is to give flexibility for the customer to pick appropriate levels and doesn't have to invest in customizing the product.

Below are the different groupings of aggregations. See "Aggregates List" for additional details.

- As-Was aggregates
- As-Is aggregates
- As-Was Corporate aggregates
- As-Is Corporate aggregates
- Season aggregates

Even though there are different flavors of aggregations based on As-Is and As-Was, there will be few aggregate tables which will be commonly used for both As-Is and As-Was. That is because, As-Is and As-Was differentiation is only across Product and Organization Hierarchy. If the aggregation for a fact table is based on Time dimension or any dimension other than Product and Organization, then that aggregate table can be used for both As-Is and As-Was. For example, since the

W_RTL_SLS_IT_LC_WK_A table is at Item and Location, it can used for either As-Was, As-Is, or both.

When the aggregations happen on any level of the Product or Organization hierarchy, there will be separate aggregates for As-Is and As-Was. For example, the W_RTL_SLS_SC_LC_DY_A and W_RTL_SLS_SC_LC_DY_CUR_A aggregates are on the subclass level of product dimension. The W_RTL_SLS_SC_LC_DY_A aggregate is for As-Was and the W_RTL_SLS_SC_LC_DY_CUR_A aggregate is for As-Is. All the As-Is aggregates are suffixed by 'CUR_A', which means 'Current.'

> **Note:**
>
> With the exception of a Corporate aggregate in Sales, Retail Insights out of the box does not have any aggregates across the Organization Dimension.

When the aggregation is only on a level from Product or Organization dimension then those are referred as Corporate Aggregates. These kinds of aggregates are very useful when reporting is done on any level of Product and Calendar hierarchy or Organization and Calendar hierarchy. For the list of this type of aggregates for every fact area see "Aggregates List". Corporate aggregates are also classified into As-Is and As-Was because they need to be processed separately to capture the current as opposed to historical parent information.

Season aggregates are useful to do reporting specific to Season dimension. All the aggregates on Season can be used for both As-Is and As-Was.

For each group of aggregations, as mentioned above, there is a mandatory aggregate table that needs to be used for other selections of the aggregates and that can be identified in the FlexAggregates document with the highlighted text.

For example, if the business only needs As-Is, the following points need to be considered:

1. Get the general usage patterns of the data and aggregation ratio. Based on that, select the list of aggregate tables. This may not be accurate for the first time but can always be changed over a period of time based on the usage and the changing data.
2. Ensure that you disable/freeze all the As-Was aggregate jobs and some of the As-Is aggregates which were not selected, in ODI and disable the same in Oracle Analytics as well. See the *Oracle Retail Insights Operations Guide* for more information.

ETL Programs Performance

Setting ETL Program Multi-threading

Retail Insights base fact load programs can be configured to run using multiple threads. The default number of threads for these programs is set to one and can be configured based on requirements. For additional information on how multi-threading works, see the Program Overview chapter of the *Oracle Retail Insights Operations Guide*.

1. Finalize the multi-threading strategy for the base fact extract programs.
2. Number of threads for each program may vary based on the data volume that program handles and resource availability. Different thread numbers should be tested by clients during implementation to achieve optimal results from multi-threading.

3. In the C_ODI_PARAM table, update the value of the PARAM_VALUE column to the desired number of threads. This applies to all records with the value 'LOC_NUM_OF_THREAD' in the PARAM_NAME column and the name of the program that requires multi-threading set in the SCENARIO_NAME column. See an example below for scenario named SIL_Test, where the desired number of threads needs to be set to 2 from 1 (default).

   ```
   UPDATE C_ODI_PARAMSET PARAM_VALUE = 2WHERE PARAM_NAME =
   'LOC_NUM_OF_THREAD'AND SCENARIO_NAME = 'SIL_Test'
   ```

4. If the number of thread required is more than 10, you need to modify the DDL for intermediate temp tables used by the ODI scenario. DDL changes require adding extra partitions to hold the data. The number of partitions on the intermediate temp table must be the same or higher than the required number of threads (which is the value for LOC_NUM_OF_THREADS set in the previous step).
5. The value setup in the C_ODI_PARAM (in step 3) should be bigger or equal than the max value of column ETL_THREAD_VAL in the staging tables. Otherwise, some records could get missing.
6. If RDE SDE programs are not used, it will be the client's responsibility to assign the data evenly across partitions in the staging tables based on the partition key column ETL_THREAD_VAL. The following is the things that need to be considered when data partition is manually made:
   - To get the most benefit of multi-threading, the data in the staging tables should be evenly partitioned by column ETL_THREAD_VAL.
   - Records with same location (store or warehouse) should have same ETL_THREAD_VAL, otherwise, unique constrain could be violated.

ODI Configuration

ODI must be configured prior to implementing Retail Insights. See the *Oracle Retail Insights Installation Guide* for details on configuring ODI in an on-premise environment. In the cloud, ODI is managed by Oracle and this documentation is informational only.

ETL Batch Scheduling

- Set up the proper dependencies between the applications to ensure resources are fully utilized, which helps the nightly batch finish earlier.
- Retail Insights load programs (SIL programs) must not wait for all the extraction programs (sde) to finish before starting. Some of them can start executing as soon as the corresponding staging table is populated. For more information on setting up dependencies, refer to the *Oracle Retail Insights Operations Guide*.
- Allocate resources to the most important batch jobs (ones that populate the tables) that support your critical reports (the reports you need first thing in the morning). You can assign job priority in most batch scheduling tools.
- Ensure that your source applications batch is optimized. Retail Insights runs towards the end of the nightly batch. Retail Insights jobs are often the last jobs to start due to the dependencies on the source system jobs, so Retail Insights is often the last to finish. Optimizing the source applications batch helps Retail Insights jobs start earlier.

Additional Considerations

- Sort the W_RTL_INV_IT_LC_G table data after the data is seeded for the first time to improve ETL performance.
- In a production environment, fact tables with large data volume can be created with the No Logging option. This improves the ETL performance and can be implemented on a case by case basis.

Report Design

Report design can affect the performance of a report. While creating custom reports, refer to the following guidelines:

- Report developers should be trained in Oracle BI to learn how to design reports in the most optimal manner.
- Design reports at the highest level possible and allow drill down to more detailed levels when required.
- Design reports in a manner that multiple users can utilize a single report output rather than multiple users running the same report. A best practice is to run one report and distribute that report to multiple users. For more information on how to distribute reports, refer to the Delivering Content chapter of the *Oracle Business Intelligence Enterprise Edition User Guide* and the Configuring Oracle BI Scheduler chapter of the *Oracle Business Intelligence System Administrator's Guide*.
- Do not design reports to request data at a level lower than the minimum level that a metric can be reported. In addition, drilling must not be performed at these levels. This ensures that reports do not produce misleading or invalid results. For example, reports must not be designed to request planning data at the item level because planning data is only available at the subclass level and above.
- Evaluate and purge reports periodically to eliminate any outdated or duplicate reports.
- Design reports to use the least amount of fact areas necessary. This reduces the number of fact table joins and in turn reduces the risk of poor report performance. For example, a best practice is not to design a single report with all sales, inventory, pricing and cost metrics, as this report will perform poorly due to joins on big fact tables. In this type of scenario, try creating separate reports with one or two fact areas on the report at a time and combining the results after these reports have run successfully.
- Design reports with the least number of metrics necessary.
- Schedule reports according to priority. This ensures that critical reports are available when needed. For more information on how to schedule reports, refer to the Configuring Oracle BI Scheduler chapter of the *Oracle Business Intelligence System Administrator's Guide*.

Additional Factors

Decision support queries sometimes require retrieval of large amounts of data. The Oracle BI server can save the results of a query in cache files and then reuse those results later when a similar query is requested. Using the middle-tier cache permits a query to be run one time for multiple runnings of a query and not necessarily every time the query is run. The query cache allows the Oracle BI Server to satisfy many subsequent query requests without having to

access back-end data sources (such as Oracle database). This reduction in communication costs can dramatically decrease query response time.

To summarize, query caching has the following advantages only when the same report is run repeatedly:

- Improvement of query performance
- Less network traffic
- Reduction in database processing and charge back
- Reduction in Oracle BI server processing overhead

For more details on Caching refer to the Managing Performance Tuning and Query Caching chapter in the *Oracle Analytics System Administrator's Guide*.

Partitioning Strategy

Database level table partitioning is very important for ETL batch and report performance. For more information, see Compression and Partitioning.

Data Base Configuration

Retail Insights is built on Oracle Database 12c and must be optimized and configured for a retailers' needs. Refer to the Setting up your Data Warehouse System chapter of the *Oracle 12c Data Warehouse Guide*.

Adequate Hardware Resources

ETL program and report performance are highly dependent on the hardware resources. For more information, see Setup and Configuration.

Leading Practices

Customizations

Changes and modifications to the Retail Insights delivered code or development of new code is considered customization. Retail Insights does not support custom code developed by clients unless the issue related to customization can be recreated using Retail Insights delivered objects. Customizations are also not supported in the cloud service model, so the following sections apply only to on-premise installations. Listed below are recommendations that will help you in maintaining Retail Insights code:

- Naming convention: it is recommended that you use a good and consistent naming convention when customizing Retail Insights delivered code or building new code in the Retail Insights environment.

  This strategy is helpful in identifying custom code and also helps when merging a retailer's Retail Insights repository with future releases of the Retail Insights repository. There is a possibility of losing customizations to Retail Insights provided ODI scripts or Oracle Analytics repository, if the customized code uses the same object/script names that are used by Retail Insights.

- As a best practice, keep all the documentation up-to-date for capturing any changes or new code that has been developed at a site. For example, if table structure has been customized, create or update the custom Data Model Guide with these changes.
- While customizing the rpd, do not make any changes directly on the main shipped/ original rpd. Make a copy of the original rpd and start the changes on the copied rpd which will be the modified version. This is useful while applying any patches in future releases of Retail Insights through Oracle Analytics's merge utility. For more details refer to the Managing Oracle BI Repository Files chapter of the *Oracle Analytics Metadata Repository Builder's Guide*.

ODI Best Practices

For customizations to existing ODI code or while creating new ODI code, refer to the *ODI Best Practices Guide* included with your product code.

Oracle Analytics Best Practices

- Create aliases for the objects created in the physical layer for usability purposes.
- Do not design the business layer model as a snow-flake model.
- Any level key on ident's must be set to non-drillable.
- In the presentation layer, fact folders (presentation tables) must contain only metrics and dimension folders (presentation tables) must contain only attributes.
- For a development environment, it is recommended to use a multi-user environment. For more information on setting up a multi-user environment, refer to the Completing Setup and Managing Oracle BI Repository Files chapter of the *Oracle Analytics Server Administration Guide*.

Batch Schedule Best Practices

The following best practices are recommended for Retail Insights:

Automation

The batch schedule should be automated as per the *Oracle Retail Insights Operations Guide*. Any manual intervention should be avoided. POM callbacks and external dependencies can be configured to link the RI batch with an on-premise scheduling system.

Recoverability

Set up the batch schedule in such a manner that the batch can resume from the point where it failed. In the cloud, this is managed automatically as part of the POM batch scheduler.

Retail Insights Loading Batch Execution Catch-Up

Loading batch (sil) execution catch-up can be achieved by backing up the staging table data. The following scenarios explain when users can benefit from this. This approach is considered a customization on Retail Insights programs and is not supported.

- Catch-up: When Retail Insights is not ready for implementation, users can use history data stored in the staging backup tables (explained later in this section) to catch-up on the data loading once the system is implemented or becomes available.

- Retail Insights database systems are down: When Retail Insights database systems are down, users can use history data stored in the staging backup tables (explained later in this section) to load the data once the system becomes available.

The following steps illustrate/show how the Loading Batch Execution catch-up solution works:

1. Create Retail Insights staging tables for each corresponding staging table using the DDL for the staging tables provided. For more information, see the *Oracle Retail Insights Administration Guide*.
2. Set up the Retail Insights ODI Universal Adapter programs, so they are ready to be executed against source files provided by the source system and load into the staging tables created in previous setup.
3. Create a one-to-one backup table for each staging table. This backup table uses the same DDL as the staging table along with an additional field (load_date) which can be mapped to source system business date. This date is used as a filter when the backup data is ready to be moved to the staging table.
4. Execute the Universal Adapter program to populate the staging tables created in the first step.
5. Move staging table data into backup staging table with the correct business date. By default, Retail Insights does not provide the backup table DDL or backup data population scripts.
6. Repeat the process of executing the Universal Adapter program and taking the backup to the staging backup table periodically (daily, once in two days, weekly, and so on), until the Retail Insights systems are available. Note that the staging table only contains the current business day's data, while the backup staging table contains data for all the business dates when the program was executed.
7. Once the Retail Insights systems become available, move the backup staging table data to the staging tables, one day at a time. This can be done by using 'load_date' as a filter on the source backup staging table data.
8. Once one business date data is moved to the staging table, SIL programs and corresponding PLP programs need to be executed for loading data into final data warehouse tables.
9. Repeat the process of moving data from backup staging to staging and executing SIL and PLP programs until all the data for all business dates from backup staging tables is loaded into the fact and dimension tables.

High Availability

Depending on your specific requirements and for facilitating performance improvement, a reporting mirror (exact copy of existing data warehouse) can be created. With this approach, one database can be used for ETL processes and the second database instance can be used by users for running their reports. There are several ways (database level solutions, operating system level solutions and hardware level solutions) of creating a database mirror. Consult with your IT resources or database administrator for evaluating available options.If this approach is adopted, you must run your queries from the reporting mirror area, not from core data warehouse area. Take the following into consideration:

- Consider this approach for large data warehouse implementations.

- Creating data marts can be a good option when implementing mirroring.
- Build a user notification mechanism should be built to notify users after the data has been refreshed on the mirror.

Advantages

- High availability of data warehouse. When batch is running, the users access the mirror and the only downtime is when data is copied over from the core data warehouse to the mirror.
- There are no conflicts between user queries and the ETL batch schedule.

Disadvantages

- Storage requirements are increased.
- Additional database maintenance is required.

Batch Efficiency

Keep revisiting the batch timings on a periodic basis to identify the candidates for performance improvements.

Aggregates List

The table below lists Retail Insights aggregates grouped by subject area and aggregation type.

Table 7-1 Retail Insights Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Inventory Position	W_RTL_INV_IT_LC_DY_F	W_RTL_INV_IT_LC_WK_A W_RTL_INV_IT_LC_GMH_AW_RTL_INV_IT_RG_DY_AW_RTL_INV_SC_LC_DY_A W_RTL_INV_SC_LC_WK_A W_RTL_INV_CL_LC_DY_A W_RTL_INV_CL_LC_WK_A W_RTL_INV_DP_LC_DY_A W_RTL_INV_DP_LC_WK_AW_RTL_INVAGE_DP_LC_DY_A	W_RTL_INV_IT_LC_WK_A W_RTL_INV_IT_LC_GMH_A W_RTL_INV_IT_RG_DY_CUR_A W_RTL_INV_SC_LC_DY_CUR_A W_RTL_INV_SC_LC_WK_CUR_A	W_RTL_INV_IT_DY_A W_RTL_INV_IT_WK_A W_RTL_INV_SC_DY_A W_RTL_INV_SC_WK_A	W_RTL_INV_IT_DY_A W_RTL_INV_IT_WK_A W_RTL_INV_SC_DY_CUR_A W_RTL_INV_SC_WK_CUR_A	

Table 7-1 (Cont.) Retail Insights Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Inventory Receipts	W_RTL_INVRC_IT_LC_DY_F	W_RTL_INVRC_IT_LC_WK_A W_RTL_INVRC_SC_LC_DY_A W_RTL_INVRC_SC_LC_WK_A W_RTL_INVRC_CL_LC_WK_A W_RTL_INVRC_CL_LC_DY_A W_RTL_INVRC_DP_LC_DY_A W_RTL_INVRC_DP_LC_WK_A W_RTL_INVRC_ID_LC_DY_AW_RTL_INVRC_ID_LC_WK_AW_RTL_INVRC_IL1_LC_DY_A W_RTL_INVRC_IL1_LC_WK_A	W_RTL_INVRC_SC_LC_DY_CUR_A W_RTL_INVRC_SC_LC_WK_CUR_A W_RTL_INVRC_IT_LC_WK_A W_RTL_INVRC_ID_LC_DY_AW_RTL_INVRC_ID_LC_WK_AW_RTL_INVRC_IL1_LC_DY_AW_RTL_INVRC_IL1_LC_WK_A	W_RTL_INVRC_IT_DY_A W_RTL_INVRC_IT_WK_A W_RTL_INVRC_SC_DY_A W_RTL_INVRC_SC_WK_A	W_RTL_INVRC_IT_DY_A W_RTL_INVRC_IT_WK_A W_RTL_INVRC_SC_DY_CUR_A W_RTL_INVRC_SC_WK_CUR_A	W_RTL_INVRC_IT_DY_SN_A W_RTL_INVRC_IT_LC_DY_SN_A W_RTL_INVRC_IT_LC_WK_SN_A W_RTL_INVRC_IT_WK_SN_A
Inventory Adjustment	W_RTL_INVADJ_IT_LC_DY_F	W_RTL_INVADJ_IT_LC_WK_A W_RTL_INVADJ_SC_LC_DY_A W_RTL_INVADJ_SC_LC_WK_A	W_RTL_INVADJ_IT_LC_WK_A W_RTL_INVADJ_SC_LC_DY_CUR_A W_RTL_INVADJ_SC_LC_WK_CUR_A			W_RTL_INVADJ_IT_LC_DY_SN_A
Inventory Transfer	W_RTL_INVTSF_IT_LC_DY_F	W_RTL_INVTSF_IT_LC_WK_A W_RTL_INVTSF_SC_LC_DY_A W_RTL_INVTSF_SC_LC_WK_A	W_RTL_INVTSF_IT_LC_WK_A W_RTL_INVTSF_SC_LC_DY_CUR_A W_RTL_INVTSF_SC_LC_WK_CUR_A			W_RTL_INVTSF_IT_LC_DY_SN_A
Return to Vendor	W_RTL_INVRTV_IT_LC_DY_F					

Table 7-1 (Cont.) Retail Insights Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Unavailable Inventory	W_RTL_INVU_IT_LC_DY_F	W_RTL_INVU_IT_LC_WK_A W_RTL_INVU_SC_LC_DY_A W_RTL_INVU_SC_LC_WK_A	W_RTL_INVU_IT_LC_WK_A W_RTL_INVU_SC_LC_DY_CUR_A W_RTL_INVU_SC_LC_WK_CUR_A			
Markdown	W_RTL_MKDN_IT_LC_DY_F	W_RTL_MKDN_IT_LC_WK_A W_RTL_MKDN_SC_LC_DY_A W_RTL_MKDN_SC_LC_WK_A W_RTL_MKDN_CL_LC_DY_A W_RTL_MKDN_CL_LC_WK_A W_RTL_MKDN_DP_LC_DY_A W_RTL_MKDN_DP_LC_WK_A W_RTL_MKDN_ID_LC_DY_AW_RTL_MKDN_ID_LC_WK_AW_RTL_MKDN_IL1_LC_DY_AW_RTL_MKDN_IL1_LC_WK_A	W_RTL_MKDN_SC_LC_DY_CUR_A W_RTL_MKDN_SC_LC_WK_CUR_A W_RTL_MKDN_CL_LC_DY_ CUR_A W_RTL_MKDN_CL_LC_WK_ CUR_A W_RTL_MKDN_DP_LC_DY_ CUR_A W_RTL_MKDN_DP_LC_WK_ CUR_A W_RTL_MKDN_IT_LC_WK_A W_RTL_MKDN_ID_LC_DY_A W_RTL_MKDN_ID_LC_WK_A W_RTL_MKDN_IL1_LC_DY_A W_RTL_MKDN_IL1_LC_WK_A	W_RTL_MKDN_IT_DY_A W_RTL_MKDN_IT_WK_A W_RTL_MKDN_SC_DY_A W_RTL_MKDN_SC_WK_A W_RTL_MKDN_SC_DY_CURR_A W_RTL_MKDN_SC_WK_CURR_A	W_RTL_MKDN_IT_DY_A W_RTL_MKDN_IT_WK_A W_RTL_MKDN_SC_DY_CURR_A W_RTL_MKDN_SC_WK_CURR_A	W_RTL_MKDN_IT_LC_DY_SN_A W_RTL_MKDN_IT_LC_WK_SN_A W_RTL_MKDN_IT_DY_SN_A W_RTL_MKDN_IT_WK_SN_A
Net Cost	W_RTL_NCOST_IT_LC_DY_F			W_RTL_NCOST_IT_DY_A	W_RTL_NCOST_IT_DY_A	

Table 7-1 (Cont.) Retail Insights Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Net Profit	W_RTL_NPROF_IT_LC_DY_F	W_RTL_NPROF_IT_LC_WK_A W_RTL_NPROF_SC_LC_DY_A W_RTL_NPROF_SC_LC_WK_A W_RTL_NPROF_CL_LC_DY_A W_RTL_NPROF_CL_LC_WK_A W_RTL_NPROF_DP_LC_DY_A W_RTL_NPROF_DP_LC_WK_A	W_RTL_NPROF_IT_LC_WK_A	W_RTL_NPROF_IT_DY_A W_RTL_NPROF_IT_WK_A W_RTL_NPROF_SC_DY_A W_RTL_NPROF_SC_WK_A	W_RTL_NPROF_IT_DY_A W_RTL_NPROF_IT_WK_A	
On Order	W_RTL_PO_ONORD_IT_LC_DY_F W_RTL_PO_ONALC_IT_LC_DY_F					
Planning	W_RTL_MFPOP_PROD3_LC3_T3_F W_RTL_PLAN1_PROD1_LC1_T1_FW_RTL_PLAN2_PROD2_LC2_T2_FW_RTL_PLAN3_PROD3_LC3_T3_FW_RTL_PLAN4_PROD4_LC4_T4_F					

Table 7-1 (Cont.) Retail Insights Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Planning Forecast	W_RTL_PLANFC1_PROD1_LC1_T1_F W_RTL_PLANFC2_PROD2_LC2_T2_F					
Pricing	W_RTL_PRICE_IT_LC_DY_F			W_RTL_PRICE_IT_DY_A	W_RTL_PRICE_IT_DY_A	
Promotion Actuals	W_RTL_PRACT_IT_LC_DY_F					
Promotion Budget	W_RTL_PRBDGT_IT_LC_F					
Promotion Forecast	W_RTL_SLSPRFC_PC_CS_DY_F W_RTL_SLSPRFC_PC_CS_WK_F					

Table 7-1 (Cont.) Retail Insights Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Sales Transaction	W_RTL_SLS_TRX_IT_LC_DY_F	W_RTL_SLS_IT_LC_DY_A W_RTL_SLS_IT_LC_WK_A W_RTL_SLS_SC_LC_DY_A W_RTL_SLS_SC_LC_WK_A W_RTL_SLS_CL_LC_DY_A W_RTL_SLS_CL_LC_WK_A W_RTL_SLS_DP_LC_DY_A W_RTL_SLS_DP_LC_WK_A W_RTL_SLS_CS_IT_LC_DY_A W_RTL_SLS_CUST_LC_DY_A W_RTL_SLS_ID_LC_DY_AW_RTL_SLS_ID_LC_WK_A W_RTL_SLS_IL1_LC_DY_AW_RTL_SLS_IL1_LC_WK_A	W_RTL_SLS_IT_LC_DY_A W_RTL_SLS_IT_LC_WK_A W_RTL_SLS_SC_LC_DY_CUR_A W_RTL_SLS_SC_LC_WK_CUR_A W_RTL_SLS_CL_LC_DY_CUR_A W_RTL_SLS_CL_LC_WK_CUR_A W_RTL_SLS_DP_LC_DY_CUR_A W_RTL_SLS_DP_LC_WK_CUR_A W_RTL_SLS_CS_IT_LC_DY_A W_RTL_SLS_CUST_LC_DY_A W_RTL_SLS_ID_LC_DY_AW_RTL_SLS_ID_LC_WK_AW_RTL_SLS_IL1_LC_DY_AW_RTL_SLS_IL1_LC_WK_A	W_RTL_SLS_IT_DY_A W_RTL_SLS_IT_WK_A W_RTL_SLS_SC_DY_A W_RTL_SLS_SC_WK_A W_RTL_SLS_SC_DY_CUR_A W_RTL_SLS_SC_WK_CUR_A W_RTL_SLS_LC_DY_A W_RTL_SLS_LC_WK_A	W_RTL_SLS_IT_DY_A W_RTL_SLS_IT_WK_A W_RTL_SLS_SC_DY_CUR_A W_RTL_SLS_SC_WK_CUR_A W_RTL_SLS_LC_DY_A W_RTL_SLS_LC_WK_A	W_RTL_SLS_IT_LC_DY_SN_A W_RTL_SLS_IT_LC_WK_SN_A W_RTL_SLS_IT_DY_SN_A W_RTL_SLS_IT_WK_SN_A
Sales Discount	W_RTL_SLSDSC_TRX_IT_LC_DY_F					

Table 7-1 (Cont.) Retail Insights Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Sales Forecast	W_RTL_SLSFC_IT_LC_DY_F W_RTL_SLSFC_IT_LC_WK_F	W_RTL_SLSFC_SC_LC_DY_A W_RTL_SLSFC_SC_LC_WK_A	W_RTL_SLSFC_SC_LC_DY_CUR_A W_RTL_SLSFC_SC_LC_WK_CUR_A	W_RTL_SLSFC_IT_DY_A W_RTL_SLSFC_IT_WK_A W_RTL_SLSFC_SC_DY_A W_RTL_SLSFC_SC_WK_A	W_RTL_SLSFC_IT_DY_A W_RTL_SLSFC_IT_WK_A W_RTL_SLSFC_SC_DY_CUR_A W_RTL_SLSFC_SC_WK_CUR_A	W_RTL_SLSFC_IT_LC_DY_SN_A W_RTL_SLSFC_IT_LC_WK_SN_A W_RTL_SLSFC_IT_DY_SN_A W_RTL_SLSFC_IT_WK_SN_A
Sales Pack	W_RTL_SLSPK_IT_LC_DY_F	W_RTL_SLSPK_IT_LC_WK_A		W_RTL_SLSPK_IT_DY_A W_RTL_SLSPK_IT_WK_A	W_RTL_SLSPK_IT_DY_A W_RTL_SLSPK_IT_WK_A	W_RTL_SLSPK_IT_LC_DY_SN_A W_RTL_SLSPK_IT_LC_WK_SN_A W_RTL_SLSPK_IT_DY_SN_A W_RTL_SLSPK_IT_WK_SN_A

Table 7-1 (Cont.) Retail Insights Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Sales Promotion	W_RTL_SLSPR_TRX_IT_LC_DY_F	W_RTL_SLSPR_PC_CS_IT_LC_DY_AW_RTL_SLSPR_PC_CUST_LC_DY_AW_RTL_SLSPR_PC_HH_WK_AW_RTL_SLSPR_PC_IT_LC_DY_AW_RTL_SLSPR_PE_CS_IT_LC_DY_A W_RTL_SLSPR_PE_CUST_LC_DY_AW_RTL_SLSPR_PE_IT_LC_DY_A W_RTL_SLSPR_PP_CS_IT_LC_DY_AW_RTL_SLSPR_PP_CUST_LC_DY_AW_RTL_SLSPR_PP_IT_LC_DY_A	W_RTL_SLSPR_PC_CS_IT_LC_DY_A W_RTL_SLSPR_PC_IT_LC_DY_A W_RTL_SLSPR_PC_CUST_LC_DY_A W_RTL_SLSPR_PC_HH_WK_A W_RTL_SLSPR_PE_CS_IT_LC_DY_A W_RTL_SLSPR_PE_CUST_LC_DY_A W_RTL_SLSPR_PE_IT_LC_DY_A W_RTL_SLSPR_PP_CS_IT_LC_DY_A W_RTL_SLSPR_PP_CUST_LC_DY_A W_RTL_SLSPR_PP_IT_LC_DY_A			
Stock Ledger	W_RTL_STCK_LDGR_SC_LC_WK_F W_RTL_STCK_LDGR_SC_LC_MH_F					
Store Traffic	W_RTL_STTRFC_LC_DY_MI_F					

Table 7-1 (Cont.) Retail Insights Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Supplier Compliance	W_RTL_SUPPCM_IT_LC_DY_F W_RTL_SUPPCMUF_LC_DY_F	W_RTL_SUPPCM_IT_LC_WK_A W_RTL_SUPPCMUF_LC_WK_A	W_RTL_SUPPCM_IT_LC_WK_A W_RTL_SUPPCMUF_LC_WK_A	W_RTL_SUPPCM_LC_WK_A	W_RTL_SUPPCM_LC_WK_A	
Supplier Invoice	W_RTL_SUPP_IVC_PO_IT_F					
Transaction Tender	W_RTL_TRX_TNDR_LC_DY_F					
Unit Cost	W_RTL_BCOST_IT_LC_DY_F			W_RTL_BCOST_IT_DY_A	W_RTL_BCOST_IT_DY_A	
Wholesale Franchise	W_RTL_SLSWF_IT_LC_DY_F	W_RTL_SLSWF_IT_LC_WK_A W_RTL_SLSWF_SC_LC_DY_A W_RTL_SLSWF_SC_LC_WK_A	W_RTL_SLSWF_IT_LC_WK_A W_RTL_SLSWF_SC_LC_DY_CUR_A W_RTL_SLSWF_SC_LC_WK_CUR_A	W_RTL_SLSWF_IT_DY_A W_RTL_SLSWF_IT_WK_A	W_RTL_SLSWF_IT_DY_A W_RTL_SLSWF_IT_WK_A	
CO Tender Type	W_RTL_CO_HEAD_TNDR_LC_DY_F					
CO Promotion Transaction	W_RTL_COPR_LINE_IT_LC_DY_F W_RTL_COPR_HEAD_LC_DY_F					

Table 7-1 (Cont.) Retail Insights Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Market Sales	W_RTL_MKTSLS_TA_CH_CNG_WK_F W_RTL_MKTSLS_TA_CH_HG_WK_F				W_RTL_MKTSLS_TA_CL_CS_QR_CUR_A W_RTL_MKTSLS_TA_CL_QR_CUR_A W_RTL_MKTSLS_TA_CMG_CS_QR_A W_RTL_MKTSLS_TA_CMG_QR_A	
Customer Order Fulfillment	W_RTL_CO_LINE_FL_IT_LC_DY_F	W_RTL_COFL_IT_LC_DY_A W_RTL_COFL_IT_LC_WK_A W_RTL_COFL_SC_LC_DY_A W_RTL_COFL_SC_LC_WK_A	W_RTL_COFL_IT_LC_DY_A W_RTL_COFL_IT_LC_WK_A W_RTL_COFL_SC_LC_DY_CUR_A W_RTL_COFL_SC_LC_WK_CUR_A	W_RTL_COFL_IT_CH_DY_A W_RTL_COFL_IT_CH_WK_A W_RTL_COFL_SC_CH_DY_A W_RTL_COFL_SC_CH_WK_A	W_RTL_COFL_IT_CH_DY_A W_RTL_COFL_IT_CH_WK_A W_RTL_COFL_SC_CH_DY_CUR_A W_RTL_COFL_SC_CH_WK_CUR_A	W_RTL_COFL_IT_CH_DY_SN_A W_RTL_COFL_IT_CH_WK_SN_A W_RTL_COFL_IT_LC_DY_SN_A W_RTL_COFL_IT_LC_WK_SN_A
Customer Order Status	W_RTL_CO_LINE_STATUS_F	W_RTL_CO_STATUS_IT_LC_CH_DY_A W_RTL_CO_STATUS_IT_LC_CH_WK_A W_RTL_CO_STATUS_SC_LC_CH_DY_A W_RTL_CO_STATUS_SC_LC_CH_WK_A	W_RTL_CO_STATUS_IT_LC_CH_DY_A W_RTL_CO_STATUS_IT_LC_CH_WK_A	W_RTL_CO_STATUS_IT_CH_DY_A W_RTL_CO_STATUS_IT_CH_WK_A W_RTL_CO_STATUS_SC_CH_DY_A W_RTL_CO_STATUS_SC_CH_WK_A	W_RTL_CO_STATUS_IT_CH_DY_A W_RTL_CO_STATUS_IT_CH_WK_A W_RTL_CO_STATUS_SC_CH_DY_CUR_A W_RTL_CO_STATUS_SC_CH_WK_CUR_A	

Table 7-1 (Cont.) Retail Insights Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Customer Order Transaction	W_RTL_CO_HEAD_LC_DY_F W_RTL_CO_LINE_IT_LC_DY_F	W_RTL_CO_IT_LC_CH_DY_A W_RTL_CO_IT_LC_CH_WK_A W_RTL_CO_SC_LC_CH_DY_A W_RTL_CO_SC_LC_CH_WK_A W_RTL_CO_CS_IT_LC_CH_DY_A	W_RTL_CO_IT_LC_CH_DY_A W_RTL_CO_IT_LC_CH_WK_A W_RTL_CO_SC_LC_CH_DY_CUR_A W_RTL_CO_SC_LC_CH_WK_CUR_A W_RTL_CO_CS_IT_LC_CH_DY_A	W_RTL_CO_IT_CH_DY_A W_RTL_CO_IT_CH_WK_A W_RTL_CO_SC_CH_DY_A W_RTL_CO_SC_CH_WK_A	W_RTL_CO_IT_CH_DY_A W_RTL_CO_IT_CH_WK_A W_RTL_CO_SC_CH_DY_CUR_A W_RTL_CO_SC_CH_WK_CUR_A	W_RTL_CO_IT_LC_CH_DY_SN_A W_RTL_CO_IT_LC_CH_WK_SN_A W_RTL_CO_IT_CH_DY_SN_A W_RTL_CO_IT_CH_WK_SN_A
Gift Card Sales	W_RTL_GCN_TRX_LC_DY_F					
Touch Point	W_RTL_CO_HEAD_TP_LC_DY_F					
Customer Loyalty	W_RTL_CUST_LYL_TRX_LC_DY_F					
Customer Loyalty	W_RTL_CUST_LYL_AWD_TRX_DY_F					

8

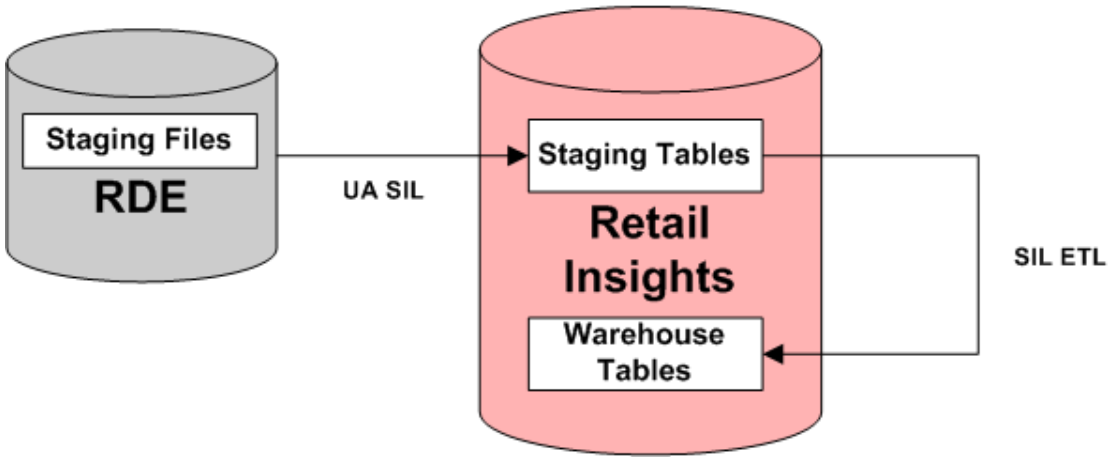
Retail Insights Universal Adapter

This chapter describes the process of implementing the Retail Insights Universal Adapter Framework. This is managed by Oracle in cloud environments and is provided as background information to understand Retail Insights operations.

Overview of Retail Insights Universal Adapter Framework

The Retail Insights BI product offering was intended to work closely with Oracle Retail's transactional schema, RMS. As such, Retail Analytics (the earlier versions of Retail Insights) shipped with source dependent extraction (SDE) routines designed to move data from RMS tables into Retail Analytics staging tables is now moved to Retail Data Extractor., The new version of Retail Insights works closely with Retail Data Extractor and processes the staging data sent in the form of flat files and loads to staging tables in RA Data Mart schema. The source independent load (SIL) moves data from staging tables into warehouse tables (see Figure 8-1).

Figure 8-1 RDE to Retail Insights Staging Data Flow



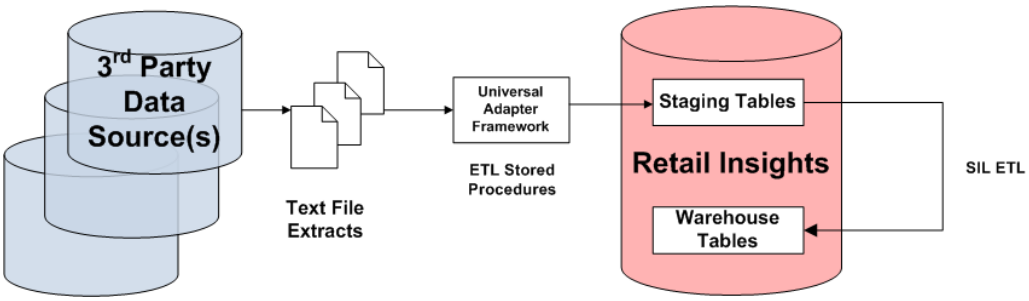
Customers who are working with third party (non-RMS) systems who wish to use Retail Insights would need to write their own ETL solution to move their data into the Retail Insights staging tables. The design of a custom ETL solution would be driven by such factors as:

- The number and nature of data sources (relational, mainframe, file-based, etc.) containing the necessary transaction data.
- The topology of the data sources. Such customers would either need to write custom SDE ETL interfaces for use with Retail Insights' ODI-based ETL system or create their own ETL logic from scratch.

The goal of the Universal Adapter Framework (UAF) is to simplify the process of moving source dependent extracts into Retail Insights staging tables for customers in a cloud/on-premise environment. The files arriving from RDE or non RMS systems should provide pipe

('|') separated value (DAT) text file extracts to be RI. All date columns should use a format of "YYYY-MM-DD;HH24:MI:SS". Once the DAT files are in place, the UAF can be used to move that data into Retail Insights staging tables through the use of Oracle sqlldr (see Figure 8-2). The control files required for sqlldr will be created automatically during the processing that is controlled in ODI.

Figure 8-2 Moving Third Party Extracts into Retail Insights Staging Tables



Benefits

Customers who elect to leverage the UAF will enjoy the following benefits:

- For customers whose third party data sources are non-relational in nature (for example, mainframe data), their development efforts only need to be focused on delivering DAT text file extracts in a pre-defined format as inputs to the UAF.
- DBlink that was used in Retail Analytics SDE programs is not required anymore. This will provide security compliance.

Universal Adapter Installation and Configuration

The installation of UAF is included in the RI standard installation. Please refer to the Oracle Retail Insights Administration Guide for UAF installation and configuration information.

Please ensure the following ODI installation files have permissions to execute the Universal adapter

1. From the ODI installation directory navigate to the ODI SDK library files and assign 775 permissions to the files mentioned below.

   ```
   Path: cd <$ODIHOME>/../../oracledi.sdk/lib/Lib/

   Command to Execute: chmod 775 <filename>
   ```

2. Replace the below filenames to assign 775 permissions to each of these.
   - os$py.class
   - stat$py.class
   - posixpath$py.class
   - warnings$py.class
   - types$py.class

- linecache$py.class

Universal Adapter Execution

Execute the Retail Insights script rtluasil.ksh and rabeuasil.ksh to run the Universal Adapter for loading. Script rabeuasil.ksh is for the target tables owned by RI batch user and script rtluasil.ksh is for the target tables owned by RI data owner.

Syntax:

```
rtluasil.ksh <Target table>
rabeuasil.ksh <Target table>
```

The Universal adapter programs accepts two types of file inputs.

- **Dat file** – This file contains the staging data that will be loaded to Fact and Dimension tables. Data files are mandatory for all the Staging tables to be loaded. In order to load legless stage tables, It is customers responsibility to generate dat files with data and place them in the "$MMHOME/data/staging"directory.
- **Ctx file** – This is an optional file and contains the metadata information that will be used to adjust ctl file generated by sqlloader in Universal adapter. This ctx file is only required if there is a mismatch in the datatypes in the source data in text files and the target loading database.

Before starting the execution download the exported zip file and extract staging data files into $MMHOME/data/staging directory.

The following is the download file process.

1. Connect to <server> port 22.
2. Log in with the SFTP User credentials.
3. Change directory to /<SFTP User>/EXPORT.
4. Extract the tar file <Merch_Extract_date>.tar into $MMHOME/data/staging directory.
5. The tar file <Merch_Extract_date>.tar can be deleted from the /<SFTP User> directory after the data files / ctx files are extracted, but Oracle recommends you archive these files for future reference and logging purposes.

Batch Logging:

The batch for Universal Adapter will have the same logging logic as other RI batch programs. The execution status can be found in RI batch maintenance table C_LOAD_DATES and ODI Operator. Besides these, Universal Adapter also provides sqlldr log files for more information of the loading in detail. The sqlldr log files can be found under $MMHOME/data/staging/log.

9

OAS Cache Usage

Caching is a feature in Oracle Analytics where a report can be cached for better performance. Enabling the cache helps to set some part of the disk space to be used for caching. Once the caching is enabled, the reports that are frequently run are stored for faster retrieval. Caching is enabled in RI cloud environments, but it requires also requires a batch program be kept enabled in your nightly batch in order to work. The feature includes:

- Setting up cache
- Clearing cache after the batch is run
- Re-caching the reports after the original cache is cleared.

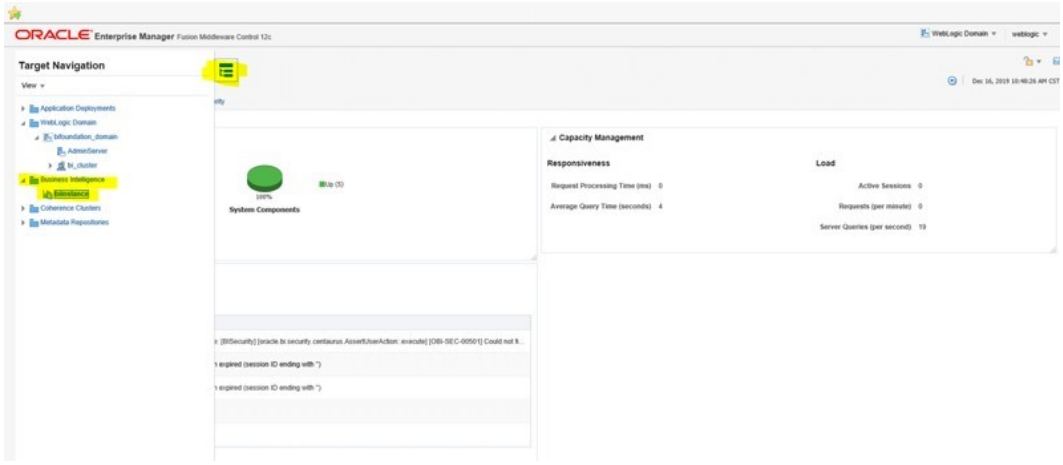
The obiee_clear_cache program is a part of the nightly batch runs and clears the cache. This will be run at the end of the batch cycle. Additional caching reports agents can be added for caching reports as additional tasks in the agent.

- Step 1: Setup Caching
- Step 2: Setup parameters for Caching in ra.env
- Step 3: Clearing Cache using the clear cache script.

Step1: Set up OAS Caching

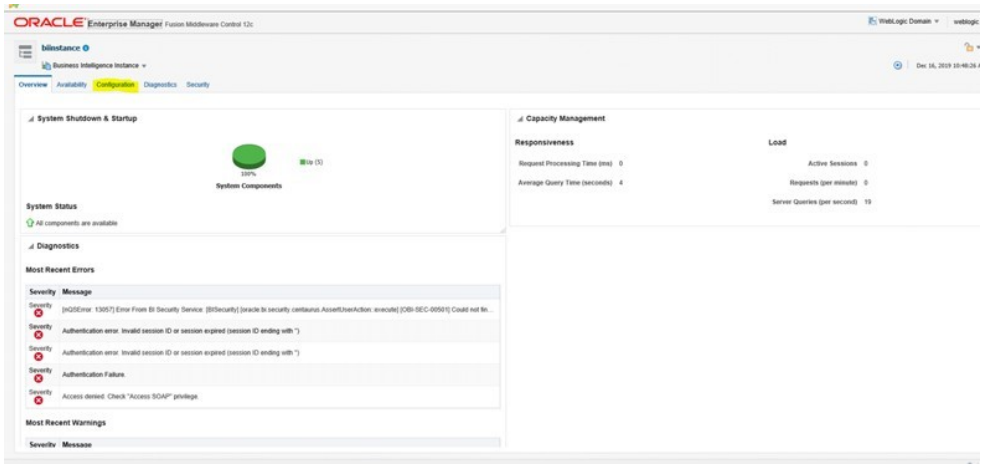
1. Go to Enterprise Manager https://nsh00amu.us.oracle.com:7002/em
2. Login with wlsadmin/weblogic userid and password
3. Click the Menu icon to open the Target Navigation menu.

Figure 9-1 Target Navigation Menu



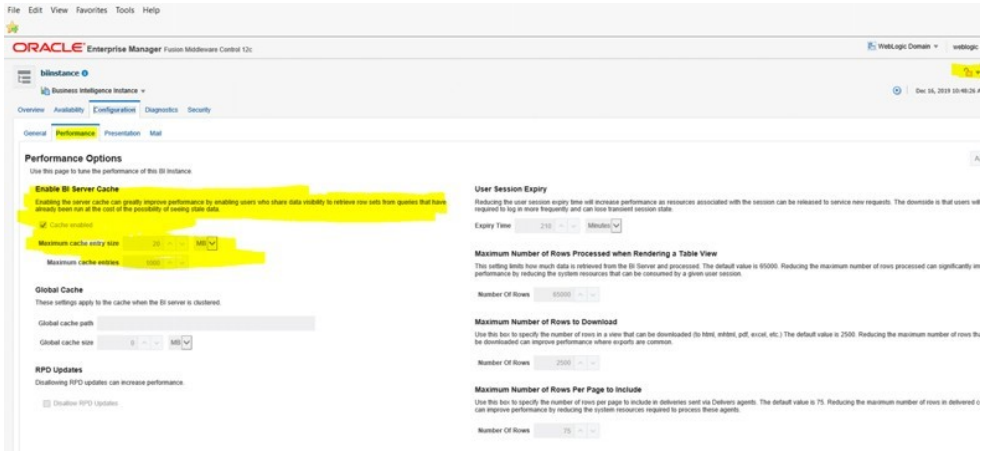
4. Expand the Business Intelligence menu item and click biinstance. The Overview tab appears.

Figure 9-2 Overview Tab



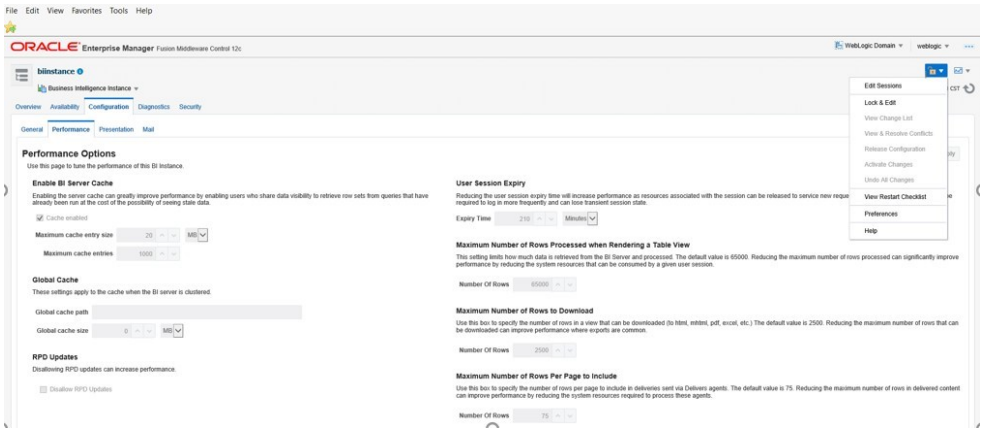
5. Click the Configuration tab.

Figure 9-3 Configuration Tab



6. Go to the top right hand corner, click Lock and Edit. Modify and then Activate after you complete editing.

Figure 9-4 Lock and Edit



Step 2: Setup Parameters for OAS Caching in ra.env

- OAS Caching also needs the entry in the wallet for rh.biuser as obiee-user-alias: Ensure that bi administrator role is available for the rh.biuser in OAS.
- Ra.env contains changes when the installer is run, following need to be validated.
  - Non Clustered Environments need to have the variable OBIEE_SEC_SCH_ SERVER and OBIEE_SEC_SCH_PORT as null. Variable OBIEE_PRI_SCH_ PORT need to be 9713 and OBIEE_PRI_SCH_SERVER should have the right name of the application server.
  - Clustered Environments need to have both OBIEE_PRI_SCH_PORT and OBI-EE_SEC_SCH_PORT as 9710. OBIEE_PRI_SCH_SERVER need to have Primary OBIEE server information and OBIEE_SEC_SCH_SERVER need to have the secondary OBIEE Server Information.

For Single Node/ Non-Clustered Environment

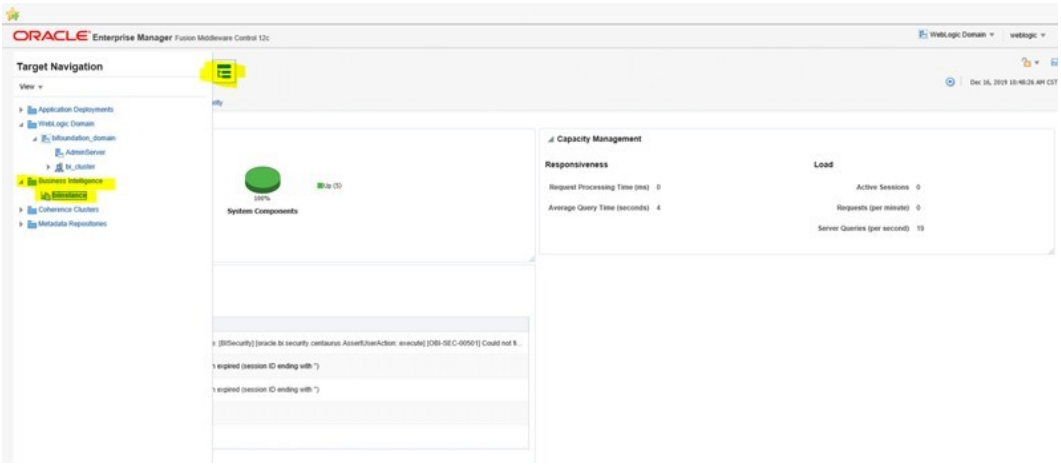
- input.obiee.pri.sch.server= Node-1-Host name
- input.obiee.sec.sch.server=
- input.obiee.pri.sch.port= 9713 (obisch1service name)
- input.obiee.sec.sch.port=

To find out the port Numbers Log into the Enterprise Manager in WLS Example of the link is:

http://nsh00amu.us.oracle.com:7001/em

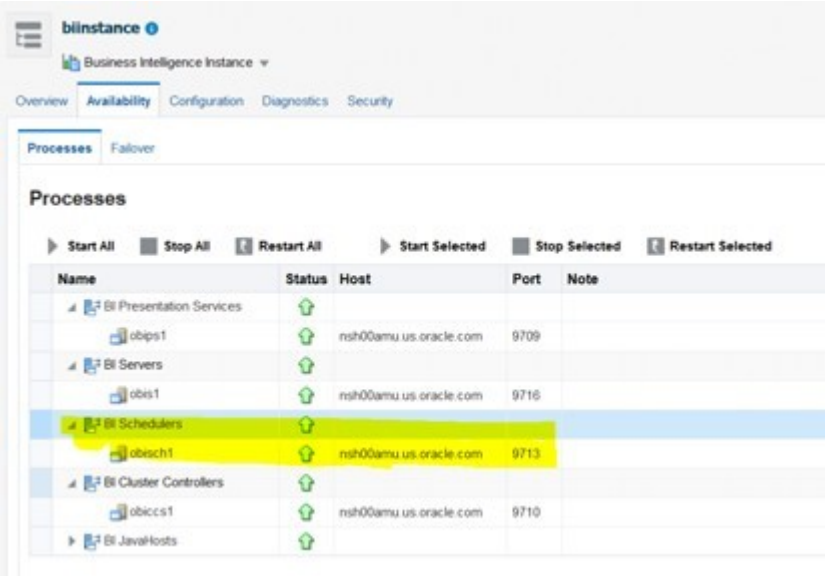
1. Go to Enterprise Manager https://nsh00amu.us.oracle.com:7002/em
2. Login with wlsadmin/weblogic userid and password
3. Click the Menu icon to open the Target Navigation menu.

Figure 9-5 Target Navigation Menu



4. Expand the Business Intelligence menu item and click biinstance. The Overview tab appears.
5. Click the Availability tab.

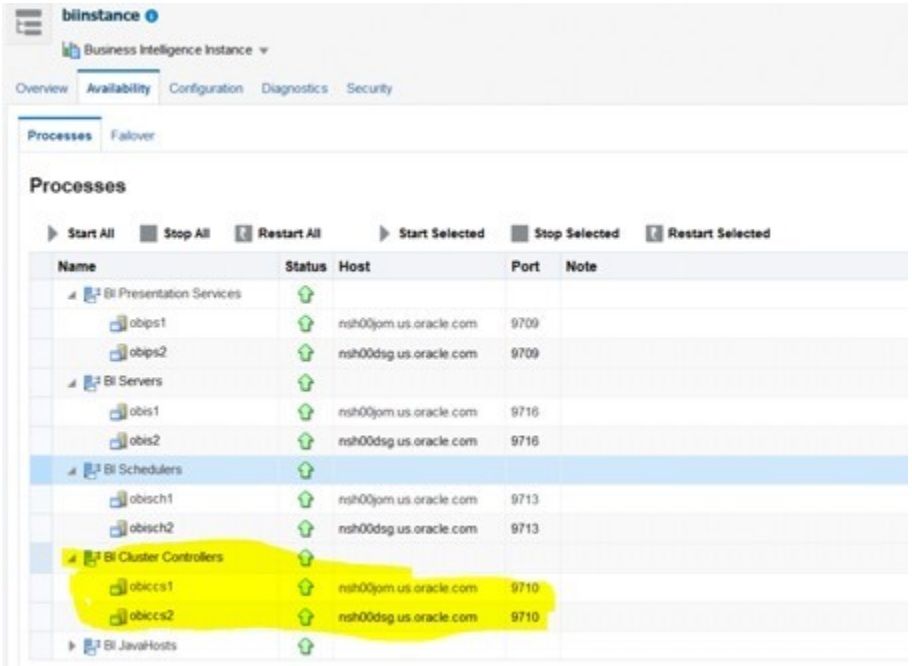
Figure 9-6 Availability Tab - Non-Clustered Environment



For Clustered Environments

- input.obiee.pri.sch.server= Node-1-Host name
- input.obiee.sec.sch.server= Node-2-Host name
- input.obiee.pri.sch.port= 9710 (obiccs1 service port)
- input.obiee.sec.sch.port= 9710 (obiccs2 service port)

Figure 9-7 Availability Tab - Clustered Environment

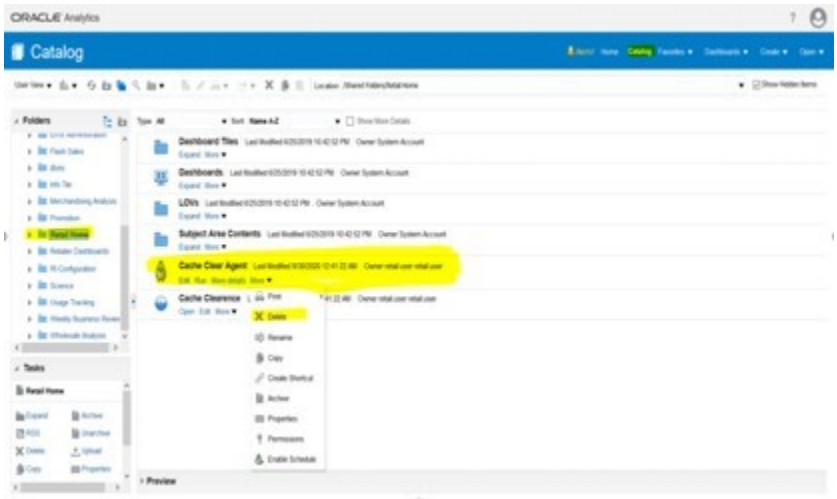


Step3: Delete and Recreate the Clear Cache Agent

Perform the following procedure to clear the cache agent deletion.

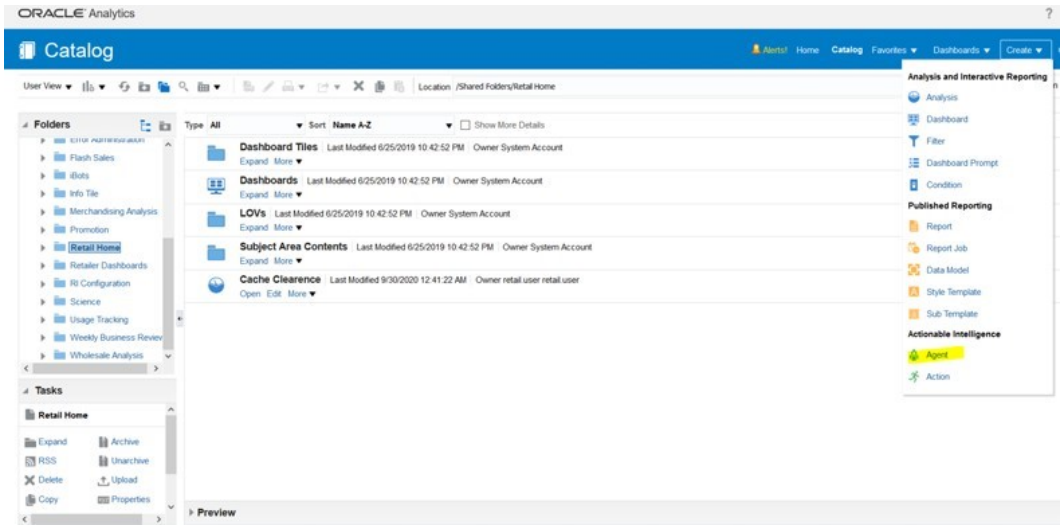
1. Navigate to Analytics URL -> go to Catalog -> Retail Home. Click the More menu and select Delete to delete the agent.

Figure 9-8 Delete the Agent



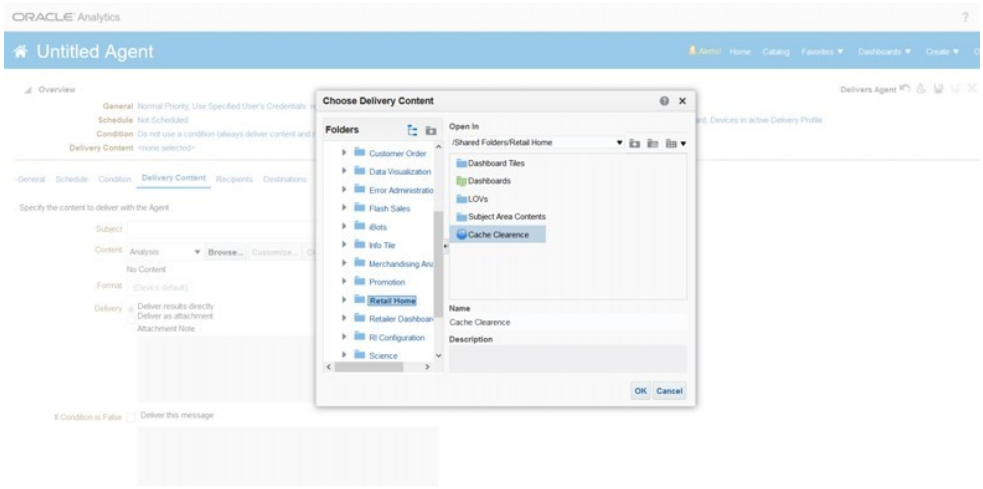
2. In the upper right of the page, click the Create menu and select Agent.

Figure 9-9 Create Agent



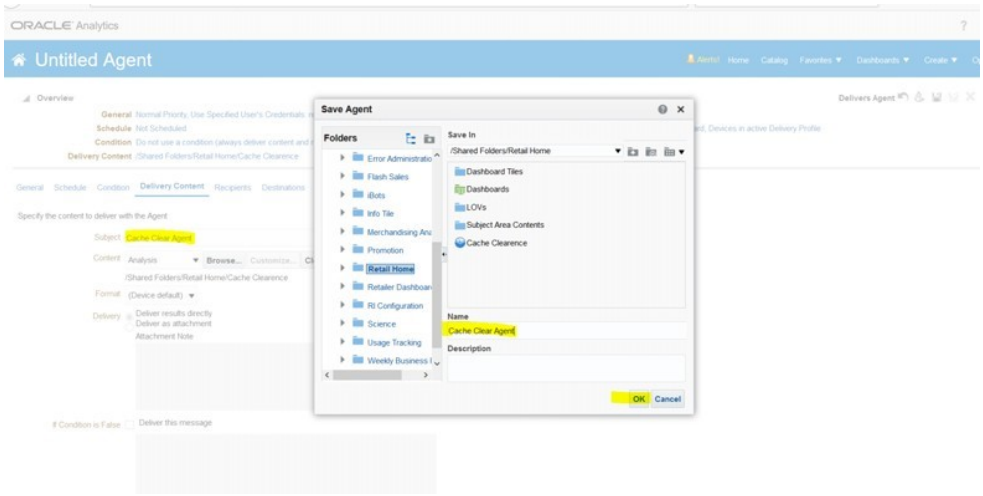
3. From the Choose Delivery Content pop-up, select Cache Clearance.

Figure 9-10 Choose Delivery Content



4. Save as Cache Clear Agent in the Shared/Retail Home Folder.

Figure 9-11 Save as Cache Clear Agent



Step 4: Clearing Cache using the Clear Cache Script

Run the following code:

```
obiee_cache_clear.sh
```

10
Frequently Asked Questions

The following issues may be encountered while implementing Retail Insights. The accompanying solutions will help you work through the issues.

Issue: Why am I getting the Login Denied error with the following message when I try to run a report using Oracle BI Presentation Services?

```
ORACLE ERROR CODE: 1017, MESSAGE: ORA-01017: INVALID USERNAME/PASSWORD; LOGON DENIED
```

Solution: Ensure that the repository connection pool has the right login credentials in the Oracle BI Administration Tool and check the tnsnames.ora file.

Issue: I am getting the following error when I performed the "Update all Row Counts" task from the Oracle BI Administration tool.

```
UNABLE TO CONNECT DATABASE USING CONNECTION POOL
```

Solution: Ensure the repository connection pool has the right login credentials in Oracle BI Administration tool or check the tnsnames.ora entry.

Issue: Why can't I see query activity at the individual user level in the NQQUERY.LOG file?

Solution: Check the logging level field in the user dialog box in the User tab. If the logging level is set to zero, the administrator may have disabled query logging. Contact the Oracle BI administrator to enable query logging.

Issue: Why is the data not loaded to the fact table, even though I have valid data in the staging table?

Solution: Data may be missing in the corresponding dimension table(s) or the transaction date is not in the active time period of the dimension.

Issue: Why is the data not loaded to the dimension table, even though I have valid data in staging table?

Solution: Parent data may be missing in the corresponding dimension table. This applies to dimensions with hierarchy.

Issue: Why is the data not loaded to the fact table, even though I have valid data in the fact staging tables and all the corresponding keys in the dimensions?

Solution: Check the effective start and end date values in the dimension tables. If any of these dimension's effective from date values is greater than the fact date value, those will not be loaded to the fact tables as those are the future dimension records.

Issue: Description of a subclass is changed from a to b in the source system but I cannot see both the records in Retail Insights after the loading process?

Solution: This type of change does not alter the relationship of subclass to any other level of the hierarchy above or below it. The record is simply updated to reflect the description change; as it is tracked as scd type 1 change. For more information, refer to the *Oracle Retail Insights Operations Guide.*

Issue: Why the load program performance is not improving even after using ODI multi-threading?

Solution: This can occur because of several reasons. Check the following settings:

- The number of threads must be appropriate for the hardware and data volume.
- The number of partitions on the intermediate temp table must be equal to or higher than the number of threads.

Issue: How do you execute the failed threads for multi-threading programs?

Solution: This can be done by using the batch log in the C_LOAD_DATES table. Table C_LOAD_DATES has a record for the execution status of each batch at thread number level. Same thread of a same batch cannot be executed twice unless the log record is deleted manually. This provides a possibility to re-execute only one thread for a case when only one thread fails and other threads complete successfully. To re-execute failed threads, the user can manually delete the threads that need to be executed and keep all other threads untouched in the C_LOAD_DATES table. Then the user can start the batch again. When the re-execution is done, the program will show errors in the UNIX console, but the threads that need to be re-executed should complete successfully. The error in the UNIX console is for the re-execution of the threads that completed successfully in the first execution, so it can be ignored.

Issue: While running the packages it is possible that there could be a scenario failure with error "Variable has no value" due to ODI out of memory.

Solution: If this error occurs, verify the values of the following two parameters are set as below (for more details refer to the *Oracle Retail Insights Installation Guide*) and regenerate the scenario that is failing.

- ODI_INIT_HEAP=256M
- ODI_MAX_HEAP=1024M

Issue: While loading data from files to RI staging tables by using Universal Adapter, there could be an error due to index (PK index) in unusable state.

Solution: This could be caused by duplicate records in the source file. Due to DIRECT load is used in the sqlldr, the PK index will be disabled when this type of error happens. The ender user can clean up the source data, re-enable the PK index on the target table (staging table), clean up records in the C_LOAD_DATES table, and then re-execute the program.