

Oracle Retail AI Foundation Cloud Services

Implementation Guide



Release 26.2.301.0

G5396-02

August 2026

ORACLE®

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, MySQL, and NetSuite 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.

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>.

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Contents

Send Us Your Comments

Preface

Audience	i
Documentation Accessibility	i
Customer Support	i
Improved Process for Oracle Retail AI Foundation Cloud Services Documentation Corrections	i
Oracle Retail AI Foundation Cloud Services Documentation on the Oracle Technology Network	ii
Conventions	ii

1 Introduction

Oracle Retail AI Foundation Cloud Services and Business Agility	1
Oracle Retail AI Foundation Cloud Services Advanced Clustering	2
Reporting and Analysis	2
Oracle Retail AI Foundation Cloud Services CDT Science and DT Science	3
Customer Decision Trees	3
Demand Transference Science	3
Oracle Retail AI Foundation Cloud Services Affinity Analysis	3
Common Workflow	3
Interacting with Oracle Retail AI Foundation Cloud Services	3
Hardware and Software Requirements	5
FAQs	6

2 Implementation Overview

Implementation Process	1
Implementation Steps	1
Configure the Application Roles and Users	1
Data Load Overview	1
Edit and Load Common Seed Data	2

Perform Attribute Preprocessing for CDT and DT, as Appropriate	3
Mandatory Configuration Parameters	3
Data Validations	5
Retail Home Integration	12

3 Customer Decision Trees

Input Data	1
Overview	1
Transactions Data Requirements	1
Attribute Data Requirements	2
Attributes with a Large Number of Values	2
Grouped vs. Raw Attribute Values	4
Attribute Splitting	4
Functional-Fit Attributes	4
CDT Hierarchy Used in the AIF Alternate Hierarchy	5
Customer Segments	5
Location Hierarchy	6
Setting Up Categories	6
Calculating Customer Decision Trees	6
Setting the Top Attribute	6
Excluding Attributes from the Calculation	7
Handling of the Brand Attribute	7
Limitations of the CDT Calculation	8
Choosing the Time Interval	8
Understanding the Filter Settings	8
Segments vs. Location	9
Setting the Escalation Path	9
How the CDT Score is Calculated	9
Understanding CDT Pruning	9
Overriding the CDT Calculation	10
Using the Calculation Stage	10
Setup Stage	10
Data Filtering Stage	10
Calculation Stage	11
Advanced Use	12

4 Demand Transference

DT and CDT	1
Demand Transference Model	1
An Example	2

Historical Similarity Data	2
Historical Sales Data	2
The Role of Attributes in Calculating Similarities	3
Attribute Data Requirements	4
Guidelines on Number of Attributes and Attribute Values	4
The Effect on Similarity Values	5
Avoiding Attributes with Many Values	5
Functional-Fit Attributes	6
Using Null as an Attribute Value	6
The Effect of Null Attribute Values on Similarity Values	7
Categories Containing Sub-Categories	7
Customer Segments	8
Location Hierarchy	9
Setting Up Categories	9

5 Using Demand Transference

Seasonality in Historical Sales Data	1
Assortment Elasticity	1
The Importance of Assortment Changes in Historical Data	2
Estimation of Assortment Elasticity	3
The Meaning of the Possible Values of Assortment Elasticity	4
The Substitutable Demand Percentage	4
No Requirement for a Time Interval	5
Segments vs. Locations	5
Setting the Escalation Path	6
Automatic Updating	6
Avoiding Categories with Small Assortments	6
Implementing DT for Fashion Categories	7
Proper Level for Fashion Categories	7
Seasonality (Life Cycle) Considerations	7

6 Advanced Clustering

Overview	1
Data Requirements	1
Multiple Hierarchies and Level Support	2
Clustering Criteria Supported in Store Clustering	3
Attributes in Store Clustering	4
Configuration Process	5
Copy Clusters Using Like Product Mapping	6
Multiple Clustering Approach	7

New Stores or Stores with Poor History	9
Outliers	9
Export to Excel	9
Configuring Cluster Exports	10

7 Customer Segmentation

Overview	1
Data Requirements	1
Multiple Hierarchies and Support	2
Supported Segment Criteria	2
Customer Segmentation Attributes	3
Configuration Process	4
Attribute Preprocessing	5
Segmenting Approach	6
Customer Segment Store Profile Generation	6
Preprocessing	6
Customer Metrics	7

8 Affinity Analysis

Overview	1
Data Requirements	1
MBA_ARM_SRVC_LOC_STG	2
MBA_ARM_SRVC_CONFIG_STG	2
Science Algorithms/Services	3
ARM_PH	3
ARM_PH_PROMO	3
ARM_PH_CS	3
Configurations	4
Data Output	4

9 Assortment Recommender

Prerequisites	1
Producing Better Assortments	1
Run Groups and Run Frequency	2
The Run Group Parameters	2
Data Used by the Assortment Recommender	3
Halo Effects	4
Troubleshooting	4

10 Size Profiles

Overview	1
Data Requirements	1
Hierarchy Data	1
Sales Data	3
Inventory Data	3
Product Attributes	3
Product Images Data	3
Season	4
Size and Size Range Data	4
Configurations	5
Configurations of the Export File	6
Logic of Profiles Spread Down in the Export File	6
Size Profiles for Warehouse Locations	7
Merchandise and Location Scope of the Size Profile Runs	7
Data Output	8
Batch and Ad-Hoc Jobs	8

11 Portfolio Optimization

Configuration	1
Reviewing Optimization Errors	2
Exporting Portfolio Optimization Results	2
Data Requirements for Portfolio Optimization	4

12 Pre-Pack Optimization

Overview	1
Data Requirements	1
Hierarchy Data	1
Sales Data	2
Size and Size Range Data	2
Supplier Data (Optional)	2
External Assortment Plan Data (Optional)	2
Interface Name: W_RTL_PLAN5_PROD5_LC5_T5_FS.dat	2
Configurations	4
Batch and Ad-Hoc Jobs	5

13 LifeCycle Pricing Optimization

Overview	1
Project Planning	4

Functional	4
Data Interfaces	4
Forecasting Parameters	5
Optimization	5
Walkthrough	5
Security	9
Data Filtering	9
User Roles	10
Data Input Requirements	10
Hierarchy Data	10
Inventory Data	12
Price History Data	13
Retail Code	13
Retail Sales Data	14
Promotions	14
Warehouse Inventory Allocation	15
Competitor Prices	15
Product Attributes	15
Product Images Data	16
Season	16
Season Periods	16
Season Products	17
Price Ladder	17
Currency	17
Holidays	17
Pricing Product Groups	17
Budget	17
Strategy and Business Rules	18
Custom Columns	18
External Forecast Adjustment	18
Configuration and Expected Levels for Interfaces	18
Optional* Interfaces	21
Integration with Retail Pricing Cloud Service	21
POM Jobs	23
Forecasting Science	29
Parameter Estimation	29
Overview of the Parameter Estimation Stages	29
Data Preparation	30
Preprocessing	32
Elasticity	34
Seasonality	39
Promotion	41

Output and Review of Parameters	42
Day Level and Returns Metrics	42
Day of the Week Profiles	42
Returns Metrics	44
Demand Forecasting	45
New Stores	47
Optimization Science	48
Business Rules	48
Optimization Algorithm Overview	52
Model Apply	53
Objective Function	53
Constraints	54

14 Control and Tactical Center

Strategy and Policy Management	1
Forecast Configuration for MFP and AP	2
Run Type Configurations for MFP and AP to Set Up GA Runs	4
Batch and Ad Hoc Jobs for MFP and AP Forecasting	6
Forecast Configuration for IPO (DF, IO, and LAR) and LPO	13
Batch and Ad-Hoc Jobs for IPO and LPO Forecasting	17
Loading and Calculating Event Flags in AIF	23
Loading Out-of-Stock and Outlier Indicators Through the RAP Interface	23
Loading Out-of-Stock and Outlier Indicators Through the RDX Interface	23
Calculating Outlier Indicators in AIF	24
Loading and Calculating Like Products and Like Locations in AIF	25
1. Loading Like Products and Like Locations Through the RAP Interface	25
2. Loading Like Products and Like Locations Through the RDX Interface	25
3. Calculating Like Products in AIF	25
4. Combination of Loading and Calculating Like Products in AIF	27
Item Replacement / Supersession in AIF	27
Workflow for IPO-DF Implementation	28
AIF-IPO-DF Workflow Example	29
Using the Add Multiple Run Types Feature	34
Building an Alternate Hierarchy in AIF Applications	35
Using Alternate Hierarchy to Set Up Run Types	38
Custom Jobs Through Innovation Workbench (IW)	38
Purge Forecast Run and Run Type Data	40
Configurations in AIF to Generate Forecasts for FACT Measures	40
Workflow Example for Generating Forecast for FACT Measures	43
Data Requirements for AIF	45
Data Definitions for Core Fact Files	45

Data Requirements for Various Forecast Methods	51
Data Correction and Reload Process	55
AIF Questions	57

15 Inventory Planning Optimization

Overview	1
Inventory Planning Optimization Runs	1
Inventory Planning Optimization UI Workflow	2
Security	3
Data Load Requirements	3
RAP Foundation Data	4
RSP Foundation Data (RSE_% Tables)	5
Inventory Planning Optimization Data (IO_% Tables)	5
Data Input Requirements	6
Hierarchy Data	6
Retail Sales Data	7
Inventory Data	7
Product Attributes	7
Product Images	7
Replenishment Attributes	8
Price and Cost	8
Season	8
Global Configurations	8
IPO Integration with RMS	11
Forecast Engine Setup for IPO	13
Batch and Ad Hoc Jobs for IPO	14

16 Inventory Planning Optimization-Inventory Planning

Security	1
Requirements	1
Batch Processing	1
Batch Monitoring for IPO-IP	2

17 Configuration

User Interface Authentication and Authorization	1
User Management Configuration: Configuring Users and Roles	1
User Roles	1
Configuration	2
RSE_CONFIG Table	2

Generic Configuration	3
Advanced Clustering Configurations	5
Customer Decision Tree Configurations	8
Demand Transference Configurations	11
Returns Logistics Configurations	14
Affinity Analysis Configurations	15
Resource Bundles	18
Manage Notebooks	18
Configurations for Loading Fact Measures into AIF	18
Internationalization	21
Configuration Updates	21
Configuration Tables	22
Email Notification Configuration	24

18 Attribute Processing

Attribute Preprocessing	1
Process Overview	1
Product Attribute Loading	3
Introduce New Attribute	4
Determine the Attribute Source and Define in the Tables	4
W_PRODUCT_D or W_PRODUCT_ATTR_D	4
W_RTL_ITEM_GRP1_D	4
Populate RSE_PROD_ATTR_GRP_VALUE_STG Interface	4
Populate RSE_PROD_ATTR_VALUE_XREF_STG Interface	6

19 AI Foundation Web Services

AI Foundation Web Services	1
Authentication and Authorization	1
Summary of Web Services	2
Access to RSE_CONFIG Table	2
Access to RSE_CONFIG_CODE Table	2
Advanced Clustering Export	3
Customer Segment Export	4
Retail Science Integration with XStore	6
Endpoint	6
Authentication	6
Retail Science and Retail Insight Integration	7
Configuration	7
POM Server Detail Configuration	7

20 Batch Processing

Overview	1
Incremental Exports	1
Batch Process Flow	1
Configuring Additional Data Files	3
File Transmissions	4

21 Innovation Workbench

Components	1
Retailer Workspace Schema	2
Oracle APEX	2
Workspace	2
Users and Roles	2
Oracle Advanced Analytics	5
Oracle Data Mining	6
How to Invoke Oracle Data Mining	6
Notebooks	9
Scheduling Innovation Workbench Python Notebook	14
IDCS Setup	14
Scheduling Jobs	15
REST API Documentation	15
Example	15
Notebook and Paragraph IDs	17
Considerations	17
REST API	18
Restful Service	18
Integration	21
Custom Data Loads	21
Required Files	21
Zip File Contents	22
Context File Details	22
Data Load Feedback	23
Invoking the Data Load	23
Custom Data Exports	23
Required Configuration	23
Invoking the Export	24
Sample Extensibility Use Case	24
Troubleshooting	25

DBMS Scheduler	26
Schema Objects	27

22 Process Orchestration and Monitoring

Batch Administration	1
Batch Monitoring	1
Monitoring a Batch Cycle	2
Triggering Process from POAM	2
Running a Nightly Batch Process	2
Monitoring the Process Executions	3
Error Handling	4

23 Targeted Items Recommendation

Attribute Similarity Based TIR	1
Attribute Similarity Science	1
Role of Attributes in Calculating Similarities	1
Historical Sales Data	6
Loading Product Attribute Data into AIF	7
Calculating Product-Location Range Data and Baseline Sales in AIF	10
Attribute Similarity Calculation as Part of AIF Forecasting	13
Fetching Similar Items Based on Attribute Similarity Calculated in AIF	16

24 FAQs

General Questions	1
DT Questions	1
Attribute Processing Questions	3
Lifecycle Pricing Optimization Questions	4

A Appendix: Innovation Workbench Workshop

Overview	A-1
Lab Innovation Workbench	A-5
Lab Analysis	A-8
Browse Existing Data	A-8
Using ReST to Load Data	A-11
Other Methods for Loading Data	A-15
Explore Data Analysis	A-17
Review Customer Behavior Analysis using Text Mining	A-17
Innovate	A-26

Churn Analysis using ODM	A-26
Decision Tree Model Details	A-29
Display Decision Tree Rules	A-35
Lab Share Insights - ReST API	A-41
Lab Share Insights - Task Navigation	A-44
Database Tools	A-46
How to Allocate Privileges	A-46
How to Execute a Job using DBMS Scheduler	A-47

B Appendix: Option Forecast Details

Life Cycle	B-1
Optimize History	B-4
Attribute Weights	B-6
Incrementality	B-6
Sales Potential	B-7

Glossary

Send Us Your Comments

Oracle Retail AI Foundation Cloud Services Implementation Guide, Release 26.2.301.0

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

Oracle Retail AI Foundation Cloud Services Implementation Guide provides detailed information useful for implementing and configuring the application. It helps you to understand the behind-the-scenes processing of the application.

Audience

This document is for users and administrators. This includes merchandisers, buyers, business analysts, implementation team partners, the ORC, and administrative personnel.

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 AI Foundation Cloud Services Documentation Corrections

To more quickly address critical corrections to Oracle Retail AI Foundation Cloud Services documentation content, Oracle Retail AI Foundation Cloud Services documentation may be republished whenever a critical correction is needed. For critical corrections, the republication

of an Oracle AI Foundation Cloud Services document may at times not be attached to a numbered software release; instead, the Oracle Retail AI Foundation Cloud Services 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 AI Foundation Cloud Services documentation is available on the Oracle Technology Network at the following URL:

<http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html>

An updated version of the applicable Oracle Retail AI Foundation Cloud Services 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 Retail AI Foundation Cloud Services Documentation on the Oracle Technology Network

Oracle Retail AI Foundation Cloud Services product documentation is available on the following web site:

<http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html>

(Data Model documents are not available through Oracle Technology Network. You can obtain these documents 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.
<i>italic</i>	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

The Oracle Retail AI Foundation Cloud Services combines AI, machine learning, and decision science with data captured from Oracle Retail AI Foundation Cloud Services SaaS applications and third-party data. The unique property of these learning-enabled applications is that they detect trends, learn from results, and increase their accuracy the more they are used, adding massive amounts of contextual data to obtain a clearer picture on what motivates outcomes.

The Oracle Retail AI Foundation Cloud Services are comprised of the following Cloud Services:

- Oracle Retail AI Foundation Platform Cloud Service
- Oracle Retail AI Foundation Lifecycle Pricing Optimization Cloud Service
- Oracle Retail AI Foundation Inventory Planning Optimization Cloud Service

The Oracle Retail AI Foundation Platform Cloud Service provides retailers with a data science toolkit that supports specific use-cases in planning, operations and execution and can be expanded to support broader retail uses. This includes Advanced Clustering, Customer Segmentation, Demand Transference, and Customer Decision Tree capabilities, as well as the Binning and Innovation Workbench capabilities.

The Oracle Retail AI Foundation Lifecycle Pricing Cloud Service reflect the evolution of our price and promotion optimization capabilities into an integrated life-cycle price optimization offering that enables retailers to engage their customers in an omnichannel environment while maximizing profits. The modular approach to offering life cycle pricing for promotions and markdowns separate from targeted offers enables retailers to innovate at the speed of their customer, while also accounting for the maturity of loyalty data necessary for targeted offers. The combined capabilities provide the following benefits to retailers:

- Drive optimal promotion and pricing decisions for the entire product life cycle
- Engage customers with targeted and contextual offers
- Execute consistently, incorporating price and promotion plans, projected receipts, and returns.
- Simplify decision-making through high-automation, exception-driven processes and what-if optimizations
- Maximize accuracy and scale using artificial intelligence, machine learning, and optimization on the Oracle AI Foundation data science infrastructure

Oracle Retail AI Foundation Cloud Services and Business Agility

AI Retail Foundation Cloud Services are hosted in the Oracle Cloud with the security features inherent to Oracle technology and a robust data center classification, providing significant uptime. The Oracle Cloud team is responsible for installing, monitoring, patching, and upgrading retail software. Included in the service are continuous technical support, access to software feature enhancements, hardware upgrades, and disaster recovery. The Cloud Service model helps to free customer IT resources from the need to perform these tasks, giving retailers greater business agility to respond to changing technologies and to perform more value-added tasks focused on business processes and innovation.

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

Oracle Retail AI Foundation Cloud Services Advanced Clustering

Oracle Retail AI Foundation Cloud Services Advanced Clustering is an enterprise-specific clustering solution that leverages data mining capabilities to create store groupings at various product levels using multiple inputs. These inputs include performance data, product attributes, store attributes, third-party data such as demographic data as well as consumer segments. Using embedded science and automation capabilities, retailers are able to identify patterns within available data to create the necessary customer-centric and targeted clusters to be used by downstream assortment planning, allocation/replenishment, pricing, and promotions planning processes.

The store clustering process enables the creation, review, and approval of store clusters for downstream solution use, while also providing the ability to define and use clustering templates that can be specific to given product/location combinations.

Oracle Retail AI Foundation Cloud Services Advanced Clustering provides retailers with multiple clustering generation approaches and methods. These include the creation of simple, nested, and mixed attribute clusters using multiple methods, including those that support discrete and non-discrete attributes.

The types of clusters include the following:

- Performance-based clusters (Sales Revenue, Sales Units, Gross Profit%, and so on)
- Product attribute-based clusters (Brand, Color Family, Price Band, and so on)
- Location attribute-based clusters (Store Size, Climate, Population Size, and so on)
- Consumer profile-based clusters (Consumer Segment Profiles)

In addition to the above, users have the ability to create multiple clustering scenarios within a single cluster run. This enables the ability to leverage embedded rankings, scoring logic, as well as solution recommendations to define and approve the most appropriate clusters for use in intended planning or execution processes.

Reporting and Analysis

Users can access and review the following reporting information to drive decisions related to the clustering process.

Users can perform the following:

- Determine what categories or merchandise classifications benefit most from clustering; determine the level of product or location hierarchy at which to cluster; and determine what attributes should be leveraged.
- Analyze details related to the available cluster recommendations, assessing areas such as cluster composition, performance, and attributes, as well as store level scores (in relation to total clusters).
- Review cluster scenario comparison features, visually assessing differences between the respective store cluster details.

Oracle Retail AI Foundation Cloud Services CDT Science and DT Science

This section describes CDT Science and DT Science.

Customer Decision Trees

Oracle Retail AI Foundation Customer Decision Tree Science and Demand Transference Science enable retailers to create customer segment-specific decision trees using available transaction level data. These customer decision trees are specific to their customer segments and the respective geographies they operate within, and retailers are provided a better understanding of their most important products and product attributes. Using this detailed information, the retailer is able to effectively analyze assortment coverage and identify the duplication of item types as well as prevent the removal of core items that would cause a loss of customers.

Demand Transference Science

Using Customer Decision Tree and Demand Transference Science, retailers can analyze a significant number of households (for example, in the thousands) to identify and rank which products are truly unique and whose sales are incremental, as opposed to those that can be discontinued because they are repetitive in nature and can be substituted with other products.

Understanding the incremental and substitutable sales associated with each item within an assortment, category managers can optimize the breadth of their assortments, as experienced by their customer's purchase preferences, with the optimal number of SKUs, given space constraints or financial goals.

Oracle Retail AI Foundation Cloud Services Affinity Analysis

Affinity Analysis (AA) lets retailers review the analysis about their customer market baskets. The system calculates association rules from the provided sales transaction data, which provides insight into customer shopping patterns. The process examines sales transaction data and identifies associations between different types of products. Such information can help a retailer understand that promoting one product is sufficient to help drive sales of another product, given the sales associations they exhibit.

Common Workflow

The AIF solutions have a similar workflow and user interface (UI). The workflow lets users implement new science applications using similar techniques. For example, a retailer who uses Demand Transference Science and Customer Decision Tree Science may then be able to more easily learn and use Advanced Clustering and other aspects of demand modeling. This approach lowers the future total cost of implementing various science applications.

The *Oracle Retail AI Foundation Cloud Services User Guide* provides details about using these applications.

Interacting with Oracle Retail AI Foundation Cloud Services

The following connection channels are used for interaction with the AIF:

Browser-Based

The application is accessed through a URL. The user is authenticated in order to gain access to the application. Access rights are controlled by the customer administrator through a Web application (Oracle Access Manager). Note that a role-based security policy is used. This allows the administrator to specify which applications and the tasks associated with those applications are accessible to which users.

Bulk Data Movement

Some of the AI Foundation integrations involve flat files as inputs or outputs to the process. To support uploading and downloading files in a Next Gen implementation, a bucket in Oracle Cloud Infrastructure Object Storage is created for each customer environment.

Oracle Cloud Infrastructure Object Storage service is an internet-scale, high-performance storage platform that offers reliable and cost-efficient data durability. Buckets are logical containers for storing objects. Any type of data, regardless of content type, is stored as an object. An object is composed of the object itself and metadata about the object.

Access to the bucket is through a pre-authenticated request (PAR), which is a URL that requires no further authentication to use to upload or download files to the bucket. To retrieve a PAR, you must use the File Transfer Service (FTS).

A scheduled ETL extraction process must be used to extract the required data on the customer side and send it to the application by uploading it to Object Storage via FTS. Similarly, a schedule-based set of processes must be set up to process data coming in the opposite direction: from the application to on-premise. Note that this connection is still initiated from on-premise. Data is made available by the application for the download to on-premise location and processed further. All the necessary processes and credentials are set up during implementation.

Web Services

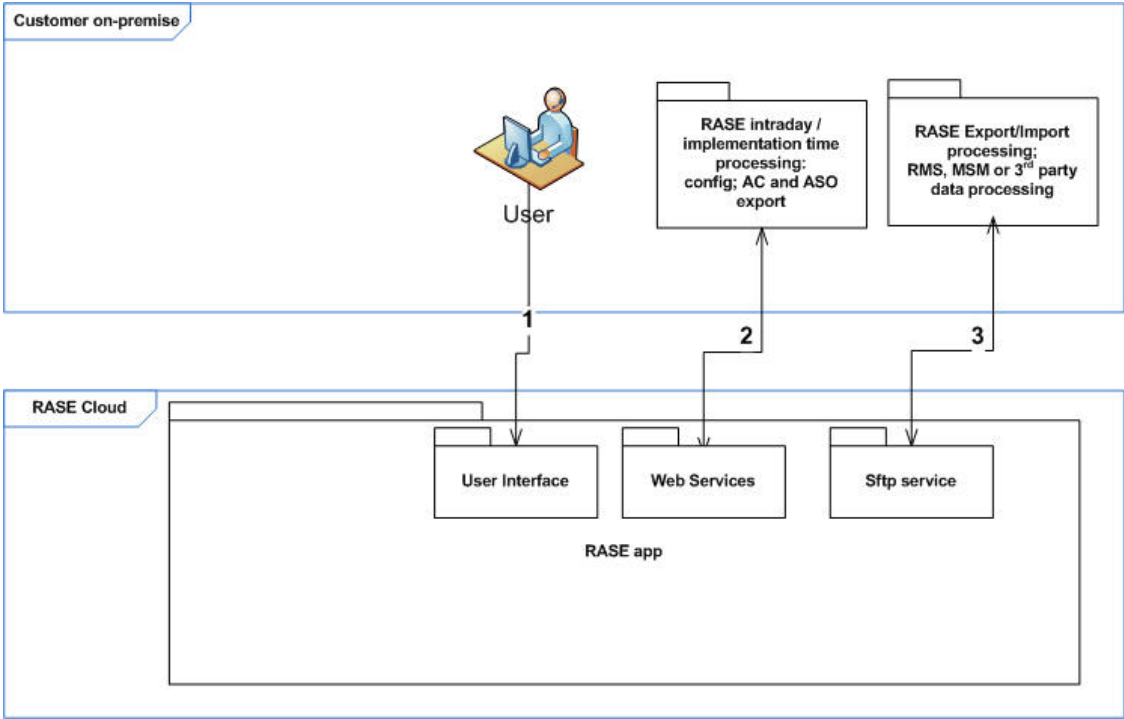
AI Foundation web services are REST-based. AI Foundation web services provide access to some of AI Foundation application data and functionality but do not fully mirror the user interface or the export and import features of the backend. They are not a replacement for bulk data export, which must still be done at a scheduled time as part of batch processing. However, access to the configuration can be used during implementation and upgrade time, while AC exports web services can serve as the means of obtaining incremental update data from a specified point in time (driven by a query parameter) as means of intra-day processing.

In the Cloud

The application processes are hooked up to a cloud scheduler to work in concert with what is sent (uploaded) from on-premise and what must be published to the outgoing SFTP directory for on-premise download.

The interactions with the application are illustrated in [Figure 1-1](#).

Figure 1-1 Interacting with Oracle Retail AI Foundation Cloud Services



Here are the major steps.

Table 1-1 Interacting with Oracle Retail AI Foundation Cloud Services

Step #	Protocol	Direction	External /Internal	Description	Type of Data Sent
1	https/443	Inbound	External (Internet)	Used by customer to communicate with the application UI, OAM (login), OIM.	Cluster, DT, CDT, space optimized results parameters
2	SQLnet/1521	App->DB	Internal	DB create, update and delete operations from the application.	Cluster parameters, query parameters, and so on.

Hardware and Software Requirements

Oracle Retail AI Foundation Cloud Services has the following requirements:

Client System Requirements

The following operating system is supported:

- Microsoft Windows 10 with Microsoft Office 2013

—

Note

AIF assumes that the retailer has ensured its Operating System has been patched with all applicable Windows updates.

The following browsers are supported:

- Mozilla Firefox
- Microsoft Edge
- Google Chrome (Desktop)

Microsoft has deprecated Internet Explorer 11 in Windows 10 and recommends using Edge as the default browser. Refer to the Oracle Software Web Browser Support Policy (<https://www.oracle.com/middleware/technologies/browser-policy.html>) for additional information.

Other Requirements

The user's source IP address must be communicated to the application cloud administration team for security purposes.

Note that all file exchange must be carried out in binary format.

FAQs

For answers to frequently asked questions, see [FAQs](#).

2

Implementation Overview

This chapter provides an overview of the implementation of Oracle Retail AI Foundation Cloud Services.

Implementation Process

This section provides details about the implementation process. It assumes that the underlying platforms have been properly implemented. This includes servers, Oracle database, RADM, and WebLogic application servers.

Note

Data interfaces are required by the application to support the various supported modules. For details about the data interface, see the following:

Oracle Retail Analytics and Planning Cloud Services Data Interfaces

Implementation Steps

The order of steps provided here is designed to simplify the process. The advanced user may be able to change the process order or skip some steps; however, that is not recommended and not documented here.

Note

See [Configuration](#) for details about application configurations that can be modified as part of a deployment.

Configure the Application Roles and Users

Before any user can log into any application, you must set up application roles, add users, and assign users to the correct roles. To do this, complete the steps described in [Configuration](#).

Data Load Overview

During an implementation of any applications, several steps are required. This section provides some details about this process.

There is a script called RSE Config and RSE Master for the common data. There are similar scripts for each application (for example, there is a CIS Config and CIS Master script). All the *Config and *Master scripts are similar in nature, so this section focuses on the common one. However, the concepts apply equally to the application-specific scripts.

Edit and Load Common Seed Data

All the applications share a set of configurable parameters that must be loaded into the RSE_CONFIG table. All have default values and are configurable and editable by the administrator. This section explains how to load and, if desired, edit these parameters.

The .ctl files for common configuration data must be edited and loaded into the staging tables. This data is common to all the applications. The application-specific .ctl files are located in their own application seed_data folders (for example, orase\installer\orase16\so\db\seed_data).

Review the .ctl files in that directory and adjust any configurations needed for the environment. Some configurations cannot be changed after the application has been used; therefore, you must carefully consider the parameters listed in [Table 2-1](#). The remainder are optional and default to reasonable values.

The following configuration parameters must be initialized during setup. The values for hierarchy level and type are recommended for any installation that integrates with the CMPO installation and must match for all installed applications.

Table 2-1 Mandatory Common RSE Database Configuration Parameters

Application	Parameter	Description	Value
RSE	CAL_PERIOD_LEVEL	This is the calendar hierarchy level that is used to drive RSE processing.	4
RSE	CHAIN_LEVEL_DESC	The description to use for any top level hierarchy element, when one must be manually created.	CHAIN
RSE	CMGRP_HIER_TYPE	The hierarchy ID to use for the CMPO. Recommendation is for a normal installation with CMPO.	1
RSE	CMGRP_LEVEL_ID	The hierarchy level ID that contains the level of the product hierarchy where the CMPO level exists (installation configuration). Recommendation is for a normal installation with CMPO.	5
RSE	MT_TZ	The time zone that is used by application server(s), that is, by the middle-tier. It must match SELECT tzname FROM V\$TIMEZONE_NAMES.	America/New_York
RSE	PRIMARY_LANGUAGE_CODE	The name of the language code to use for all RSE data sourced from the AIF data warehouse tables.	EN
RSE	RA_FISCAL_CAL_ID	The ID of the calendar to use from the AIF data warehouse (since it supports multiple calendars).	1240
RSE	TRADE_AREA_HIER_TYPE	The hierarchy ID to use for the Trade Area (installation configuration).	6
RSE	UI_TZ	The time zone for display. It must match SELECT tzname FROM V\$TIMEZONE_NAMES.	America/New_York

Perform Attribute Preprocessing for CDT and DT, as Appropriate

Product attributes are required by CDT and DT and are stored in the RADM. Attribute preprocessing is independent of the AIF applications database and happens in AIF data warehouse tables or flat files generated by the user. Once these tables and files are correct, you can load the resulting attributes in the AIFApps schema as part of the data load process.

Here are the basic attribute pre-processing steps:

1. Populate AIF data warehouse with raw attribute data using AIF DATA batch jobs.
2. Optionally, perform translation.
3. Parse.
4. Cleanse and standardize.
5. Categorize and label.
6. Define the attributes.
7. Bin and group.

For details on these steps, see [Attribute Processing](#).

Mandatory Configuration Parameters

[Table 2-2](#) contains the mandatory configuration parameters for CDT.

Table 2-2 Mandatory CDT Configuration Parameters

Application	Parameter	Description	Value
CDT	CDT_CAL_HIER_TYPE	The hierarchy ID to use for the fiscal calendar (installation configuration).	11
CDT	CDT_CAL_LEVEL_ID	The hierarchy level ID that contains the level of the calendar hierarchy that CDT operates on. (This should equate to the Week - Installation configuration).	4
CDT	CDT_CMGRP_LEVEL_ID	The hierarchy level ID that contains the level of the product hierarchy that CDTs are created for (installation configuration).	5
CDT	CDT_CUSTSEG_HIER_TYPE	The hierarchy ID to use for customer segment (installation configuration).	4
CDT	CDT_CUSTSEG_LEVEL_ID	The hierarchy level ID that contains the level of the customer segment hierarchy that CDTs are created for (installation configuration).	2
CDT	CDT_LOC_HIER_TYPE	The hierarchy ID to use for location (installation configuration).	2
CDT	CDT_LOC_LEVEL_ID	The hierarchy level ID that contains the level of the location hierarchy that CDTs are created for (installation configuration).	Best equivalent for trade area level

Table 2-2 (Cont.) Mandatory CDT Configuration Parameters

Application	Parameter	Description	Value
CDT	CDT_PROD_HIER_TYPE	The hierarchy ID to use for the CMPO (installation configuration). The recommendation is for a normal installation with CMPO.	1
CDT	DEF_NUM_WEEKS_FOR_SIMILARITY	The default number of weeks of sales transaction data to be used by the similarity process. This is used when the user does not specify time intervals.	15

[Table 2-3](#) contains the mandatory configuration parameters for DT.

Table 2-3 Mandatory DT Configuration Parameters

Application	Parameter	Description	Value
DT	AE_CALC_INT_LENGTH	The number of weeks to group together for in an interval for the AE calculation.	4
DT	CDT_SIMILARITY_AVAILABLE	Whether the CDT similarity has been made available to DT.	Y
DT	DT_CAL_HIER_TYPE	The hierarchy ID to use for the fiscal calendar.	11
DT	DT_CAL_LEVEL_ID	The hierarchy level ID that contains the level of the calendar hierarchy that DT operates on. (It should equate to week.)	4
DT	DT_CMGRP_LEVEL_ID	The hierarchy level ID that contains the level of the product hierarchy that DTs are created for.	5
DT	DT_LOC_HIER_TYPE	The hierarchy ID to use for location.	5
DT	DT_LOC_LEVEL_ID	The hierarchy level ID that contains the level of the location hierarchy that DTs are created for.	Best equivalent for trade area level
DT	DT_PROD_HIER_TYPE	The hierarchy ID to use for the CMPO.	1
DT	PR_LOC_STATUS_LAST_COMPLETED_WK	The last completed week for the SKU/Store ranging data copying.	Week ID from RSE_CAL_HIER
DT	WGT_CALC_INTERVAL_LENGTH	The number of weeks to group into an interval that is then used to perform weight calculations.	4

[Table 2-4](#) contains the mandatory configuration parameters for AC.

Table 2-4 Mandatory AC Configuration Parameters

Application	Parameter	Description	Value
CIS	CIS_DFLT_CALENDAR_HIER_TYPE_ID	the default calendar hierarchy for clustering.	11
CIS	CIS_DFLT_LOCATION_HIER_TYPE_ID	The default location hierarchy for clustering.	2
CIS	CIS_DFLT_PRODUCT_HIER_TYPE_ID	The default product hierarchy for clustering.	1
CIS	PERF_CIS_APPROACH	The approach to use for performance based clustering. The available options are CDT and DT.	CDT

Note

There are no mandatory configuration parameters for MBA.

Data Validations

AI Foundation is composed of AIF data and AIF applications. AIF applications derive information from AIF data. You can use this analysis of this data to make determine business strategies and decisions.

The following validations are performed on the AIF data.

Table 2-5 Data Validations

AIF Application Program	AIF Data Table Referenced	Validation
rse_prod_src_xref_load.ksh	- W_PRODUCT_ATTR_D - W_PRODUCT_D	- Record must be current (i.e. CURRENT_FLG is Y) - The extended hierarchy source is validated: - If the extended hierarchy source is RMS, UPC records are excluded from Merch data. - If the extended hierarchy source is different, the integration ID should be stored in either the PRODUCT_ATTR15_NAME, PRODUCT_ATTR14_NAME OR PRODUCT_ATTR13_NAME column. - Basic non-fashion and single-level fashion items, multi-level fashion items or level transaction items are extracted.

Table 2-5 (Cont.) Data Validations

AIF Application Program	AIF Data Table Referenced	Validation
rse_prod_hier_load.ksh	- W_PROD_CAT_DH - W_DOMAIN_MEMBER_LKUP_TL - W_PRODUCT_D_TL - W_PRODUCT_D	- Record must be current (i.e. CURRENT_FLG is Y) - Records should exist in RSE_PROD_SRC_XREF (i.e. prod_ext_key of this table must match the integration id). - Language or source language code should be the PRIMARY_LANGUAGE_CODE in AIF Apps, or language and source language code are the same as the domain code (MCAT). - Language or source language code should be the PRIMARY_LANGUAGE_CODE of AIF Apps configuration, or language and source language code are the same. - Product category hierarchy 5 should be equal to the row ID of W_PRODUCT_CAT_DH table. - The extended hierarchy source is validated - If extended hierarchy source is RMS, retrieve records with child level 9 with differentiator linked to the dummy parent level8. All levels (child level 9) linked to parent level 8 and parent level8 (all levels) linked to the grandparent level 7 - If extended hierarchy source is not RMS, integration ID and Product Attribute 15 should be in RSE_PROD_SRC_XREF (prod_ext_key) table
rse_load_prod_group.ksh	W_RTL_ITEM_GRP1_D	- Record must be current (i.e. CURRENT_FLG is Y) - Record must not be marked for deletion (i.e. DELETE_FLG is N) - The product group type must be ITEMLIST

Table 2-5 (Cont.) Data Validations

AIF Application Program	AIF Data Table Referenced	Validation
rse_cm_grp_xref_load.ksh	- W_RTL_CMG_PRODUCT_MTX_D - W_PROD_CAT_DH	- Data's record name is in RSE_HIER_LEVEL(DESCR) and product number is in RSE_PROD_SRC_XREF(PROD_EXT_CODE) - W_RTLCMD_PRODUCT_MTX_D product number is the same as W_PROD_CAT_DH's level 1 product hierarchy code or level 5 to 8 category code depending on RSE_HIER_LEVEL(ID). Also, the data's ROWID should be in RSE_PROD_SRC_XRED(prod_ext_id).
rse_cm_grp_hier_load.ksh	- W_RTL_CMG_PRODUCT_MTX_D - W_PROD_CAT_DH	- Data's level name should be in RSE_HIER_LEVEL(DESCR) table. - Data's product number and hierarchy level should be in RSE_PROD_SRC_XREF(PROD_EXT_CODE) table - W_RTL_CMD_PRODUCT_MTX_D product number is the same as W_PROD_CAT_DH's level 1 product hierarchy code or level 5 or level 8 category code depending on RSE_HIER_LEVEL.ID. Also, data's row ID should be in RSE_PROD_SRC_XREF.PROD_EXT_ID.
rse_loc_src_xref_load.ksh	- W_INT_ORG_DH - W_INT_ORG_ATTR_D	- Record must be current (i.e. CURRENT_FLG is Y) and not marked for deletion (i.e. DELETE_FLG is N). - Level name should be in RSE_HIER_LEVEL(DESCR) - If RSE_LOC_HIER_EXCL_FRANCHISE value is F, w_int_org_attr current flag is enabled and delete flag is disabled, and that Organization Attribute Name 23 is F

Table 2-5 (Cont.) Data Validations

AIF Application Program	AIF Data Table Referenced	Validation
rse_loc_hier_load.ksh	- W_DOMAIN_MEMBER_LKU_P_TL - W_INT_ORG_DH - W_INT_ORG_D	- Data's domain code is RTL_ORG and the domain member code is like the descr from RSE_HIER_LEVEL - If RSE_HIER_LEVEL.leaf_node is enabled, retrieve data from w_int_org_dh where scd1_wid is in w_int_org_d(scd1_wid) - Record is current (i.e. w_into_org_d.current_flg is enabled) - The w_int_org_d integration code is in the rse_loc_xec_xref(loc_ext) table - The language or source language code should be the PRIMARY_LANGUAGE_CODE of RSE config or language and source language code are the same and w_int_org_d(integration id record should be in w_int_org_d(integration id) - If RSE_HIER_LEVEL.leaf_node is disabled. - w_int_org_dh(integration_id) must be in rse_loc_src_xref(loc_ext_key) - w_domain_member_lkb_tl source language or language is the same as the RSE Config PRIMARY_LANGUAGE_CODE or language and source code are the same - The w_int_org_dg level name is in rse_hier_level(descr) - The integration id is in rse_loc_src_xref.loc_ext_key
rse_trade_area_src_xref_load.ksh	- W_RTL_TRADE_AREA_D - W_RTL_TRADE_AREA_LO C_MTX_D	- Record must be current in both tables (i.e. CURRENT_FLG is Y). - Trade area ID of the matrix table is in or equal to the rowid of w_rtl_trade_area_d table.

Table 2-5 (Cont.) Data Validations

AIF Application Program	AIF Data Table Referenced	Validation
rse_trade_area_hier_load.ksh	- W_RTL_TRADE_AREA_D - W_DOMAIN_MEMBER_LKP_TL - W_RTL_TRADE_AREA_LO_C_MTX_D	- Data of w_rtl_trade_area (integration_id) is in rse_loc_src_xref(loc_ext_key) and w_domain_member_lp_tl(domain_member_code); and the delete flag is disabled - w_domain_member_lkp_tl.domain_code is 'RTL_TRD_AREA and the language code is the PRIMARY_LANGUAGE_CODE in RSE_CONFIG. - Matrix data record (org_scd1_wid) should be in rse_loc_src_xref(loc_ext_id) and trade_area_wid matches w_rtl_trade_area_d(row_id) and record is current (i.e. current flag is enabled) - w_rtl_trade_area(integration_id) is in rse_loc_src_xref(loc_ext_id) and the record is current (i.e. current flag is enabled) and is not marked for delete (i.e. delete flag is disabled)
rse_fiscal_main_load.ksh	- W_MCAL_YEAR_D_YEAR - W_MCAL_QTR_D_QTR - W_MCAL_PERIOD_D_PERIOD - W_MCAL_WEEK_D_WEEK - W_MCAL_DAY_D_DAY - W_MCAL_YEAR_D - W_MCAL_QTR_D - W_MCAL_PERIOD_D	Values are retrieved based on the RA_FISCAL_CAL_ID defined in RSE_CONFIG.
rse_promo_src_xref_load.ksh	W_RTL_PROMO_D	Records with promo levels E, P or C are extracted.
rse_promo_hier_load.ksh	W_RTL_PROMO_D	Record exists in rse_hier_lv(descr) and promo_level is either an Event ('E'), Promotion ('P') or Component ('C').

Table 2-5 (Cont.) Data Validations

AIF Application Program	AIF Data Table Referenced	Validation
rse_custseg_src_xref_load.ksh	W_RTL_CUSTSEG_D	- Retrieves records from AIF data warehouse tables that have records in RSE_CUSTSEG_SRC_XREF. - It also retrieves all with a new custseg_hier_id (i.e. custseg_ext-key is NULL) - Record must be current (i.e. current flag is enabled) and not marked for delete (i.e. delete flag is disabled)
rse_custseg_hier_load.ksh	W_RTL_CUSTSEG_D	- Data's rowid should exist in rse_custseg_src_xref(custseg_ext_id), row_wid = -1. - Record must be current (i.e. current flag is enabled) and not marked for delete (i.e. delete flag is disabled).
rse_custseg_alloc_load.ksh	- W_RTL_CUSTSEG_ALLOC_D - W_INT_ORG_DH	- Customer segment allocation data (prod_dh_wid), and org_dh_wid should exist in w_int_org_dh (row_id) - Customer segment id should exist in rse_custseg_hier(custseg_ext_code) table - Allocation type must be CUSTOMER - Record must be current (i.e. the current flag is enabled)
rse_price_zone_grp_load.ksh	- W_RTL_CLSTR_HDR_D - W_RTL_CLSTR_GRP_D - W_DOMAIN_MEMBER_LKP_TL	- Record must be current (i.e. CURRENT_FLG is Y) and record must not be marked for delete (i.e. DELETE_FLG is N). - Cluster group type must be Zone. - Cluster group code exists in W_DOMAIN_MEMBER_LKP_TL via DOMAIN_MEMBER_CODE. - For domain member record, language or source language code should be the PRIMARY_LANGUAGE_CODE of RSE Config or, language and source language code are the same. The domain code is RETAIL_CLUSTER_GRP.

Table 2-5 (Cont.) Data Validations

AIF Application Program	AIF Data Table Referenced	Validation
rse_price_zone_load.ksh	- W_RTL_CLSTR_HDR_D - W_RTL_CLSTR_GRP_HDR_LC_D - W_RTL_CLSTR_HDR_D_CH	- Record must be current (i.e. current flag is enabled) - Cluster group wid should exist in rse_price_zone_grp(ext_id) - Cluster header wid should exist in w_rtl_clstr_hdr_d(row_wid) - Record is not marked for delete (i.e. delete flag is disabled) - Record must be current (i.e. current flag is enabled) - Cluster header wid should exist in rse_price_zone(ext_id) table - For domain member record, language or source language code should be the PRIMARY_LANGUAGE_CODE of RSE config or language and source language code are the same - Domain code is RETAIL_CLUSTER_GRP
rse_conseg_load.ksh	W_RTL_CONSUMERSEG_D	Record must be current (i.e. CURRENT_FLG is Y)
rse_fcst_sales_plan_load.ksh	- W_PROD_CAT_DH_WPCD - W_PRODUCT_D - W_INRT_ORG_DH	- For records with rse_hier_level_id less than 7: - The level_name should exist in rse_hier_level(descr) - The integration id should exist in rse_prod_src_xref(prod_ext_key) - The datasource id must be 1 - For records with rse_hier_level_id greater than or equal to 7: - Integration id should exist in rse_prod_src_xref(prod_ext_key) - The datasource id must be 1 - The w_int_org_dh(scdq_wir) record should exist in rse_loc_src_xref(loc_ext_id) and the level_name should exist in rse_hier_level(descr).

Table 2-5 (Cont.) Data Validations

AIF Application Program	AIF Data Table Referenced	Validation
rse_season_phase_item_load.ksh	W_RTL_SEASON_PHASE_IT_D	- Record must be current (i.e. current flag is enabled) - Data season wid should exist in rse_season(season_ext_id); - Phase wid should exist in rse_season_phase(phase_exts_id); - The prod_scd1_wid should exist in rse_prod_src_xref.prod_ext_id.
rse_rul_eng_pro_load.ksh	W_RTL_CLSTR_GRP_IT_D	- Data (i.e. prod_scd1_wid) should exist in rse_prod_src_xref(prod_ext_id) - The product number should exist in rsE_prod_src_Xref(prod_ext_code) - The cluster group type is 'ZONE' - Record is current (i.e. the current flag is enabled) - Record is not marked for delete (i.e. delete flag is disabled) - The ID exists in the pro_optimization_rules(price_zone_grp_id) table.

Retail Home Integration

In a Next Gen implementation, you can view usage metric values. In order to use this feature, you must ensure that the RSE_UPDATE_USAGE_METRIC process in the Process Orchestration and Monitoring application is set up as an enabled job as part of your batch schedule. This is the process that gathers the usage metric values and saves them to the subscribed modules during onboarding. See "Process Orchestration and Monitoring" for additional details on how to set up and enable the process.

3

Customer Decision Trees

This chapter provides details about the use of the Customer Decision Tree Science application.

Input Data

This section describes setting up the data that the CDT application uses to calculate CDTs.

Overview

The calculation of CDTs is based on a retailer's historical data, especially customer-linked transactions data that includes subsets of transactions from the same customer. The CDT calculation does not require any data about the customer; it does require that the transactions are flagged to indicate that they came from the same customer.

The CDT calculation uses this customer-linked transactions data to determine, for a particular category at a particular store, the switching behavior of the customer among the items in the category at that store. By seeing what fraction of all historical customers of the category consider two specific items substitutable, CDT generates a similarity for the two items, which is a number between 0 and 1 that indicates how substitutable those two items are.

It is important to have data from a large numbers of customers shopping in the category in order to be more certain of the similarity values. In general, it is not recommended to perform CDT calculation for categories where customer-linked transactions data is available only for a few hundred customers.

The CDT calculation also relies on attributes, since attributes are at the nodes of the CDT. The CDT calculation applies the similarity calculation to attribute values as well as to items in order to find the similarities between attribute values. The CDT is then generated by examining the relationships between the attribute-value similarities and the item-level similarities. So good attribute information is also important.

Notice that the CDT calculation is all within a particular category, and thus the CDT models the customer's choice process only within a category. The CDT calculation generates separate CDTs, using separate calculations, for each category that the user chooses.

The CDT calculation does not filter out the effects of promotions or price changes, because these effects can cause customers to switch to a different item. This is valuable since the basis of the CDT calculation is examining switching behavior among the customers. Generally, more switching behavior in the historical data leads to a better CDT.

Transactions Data Requirements

The historical transactions data for the CDT calculation must meet the following requirements:

Customer Linked

Since the calculation involves examining switching behavior by customers, it is necessary to identify which transactions came from the same customer. This can be done using a loyalty card or a generated customer ID. No actual information about a specific customer is required; all that is needed is a way to identify which transactions come from the same customer. Note

that it is possible to have customer IDs from a retailer where the customer ID is not that of an actual customer but rather a cashier loyalty card that was used for many different customers. These customer IDs, and their associated history, cannot be used for the CDT calculation, since the data comes from a large number of different customers. The data load for CDT automatically filters out such "fake customers."

Repeat Purchase

The category used for the calculation must be one where the typical customer performs several transactions per year. Examples include grocery items such as milk or yogurt, which are typically purchased weekly, and batteries, which are typically purchased several times per year. Item such as electronics are not appropriate, as such items may only be purchased every few years. Note that it can be possible to trade off purchase frequency and history length. It is also possible to trade off purchase frequency with the number of customers who shop in the category.

Attribute Data Requirements

The attribute values for the CDT calculation must meet the following requirements:

Set of Attributes

Each category is characterized by a unique set of attributes. These attributes differ from category to category. For example, for yogurt, the attributes might be size, flavor, brand, fat percentage, and pack size. For chocolate, the attributes might be size, brand, milk/dark, nut type, and package type. Two categories can both have brand, but that the brand attribute will have different values for each of the categories. So brand is actually a different attribute for each category.

Mapping

Each item in the category must be mapped to its set of attribute values. This information must be obtained from the retailer. Null values are acceptable as long as they are not too numerous. The CDT application can still run even if some attribute values are listed as null for some items in a category, but too many null values decrease the reliability of the generated CDTs.

Significance

The attributes for a category must be the ones that the customers actually pay attention to when shopping in the category. They are attributes that actually affect the customers' purchasing decisions.

The process of obtaining attributes for a category and performing a mapping of items in the category to attribute values is likely to require a significant amount of time and labor, even if the retailer has the information available, since this has to be done for every category.

Attributes with a Large Number of Values

A raw attribute is one that has a large number of attribute values. For example, the brand attribute for yogurt may be a list of 50 different brands at a large grocer. Using the raw attributes directly for the CDT calculation presents a problem, because each node of a CDT expands into a set of branches whose number is equal to the number of attribute values of the attribute at the node. An expansion into 50 different branches, one for each brand, is not desirable because the CDT would become too large to examine or interpret.

Such raw attributes must be turned into grouped attributes. This involves grouping the attribute values into a small number of bins. This grouping should be done in consultation with the

retailer, who may have specific requirements. For example, a retailer may want to group soft-drink brands into Brand A, Brand B, and a third group that includes all other brands.

Another approach is to divide the values into two attributes (known as attribute splitting). For example, if the color attribute has many values, the single color attribute can be divided into two attributes, with one attribute representing the primary color and the second attribute representing a modifier. The CDT application's schema directly supports attribute grouping.

Attributes with a large number of values (for example, in the hundreds) can cause the CDT Calculation Stage to require a lot of time. Here are some approaches for handling categories that have attributes with a large number of values. The retailer should help determine which approach is appropriate.

Position of the Attribute Within the Tree

Typically, an attribute with a large number of values must not be at the top or near the top of the CDT. With such a large number of values, it is unlikely customers are first selecting from among such a large number of values and then selecting from other attributes with a smaller number of values. In addition, if such an attribute were at the top of the tree, the tree would be extremely wide and shallow. It would be extremely wide because the tree would then split into as many branches as there are attribute values. If there are 100 brands, and Brand was at the top of the tree, the tree would split into 100 different branches. As a result, the CDT would not be useful to the retailer. The tree would also be quite shallow; with 100 different branches, each branch would probably have very few SKUs, and so the branch could not be expanded much further.

In such a category, customers generally first use another attribute with a smaller number of values, and then choose an attribute with a large number of values. The following example demonstrates what happens when an attribute with a large number of values is lower in the tree.

In this example, the top attribute in the tree indicates the Market Segment, so that the SKUs in the category are split into various sub-categories. The Brand has a large number of attribute values. Because the Brand is below Market Segment, the branch for each segment only has a small subset of the Brands. Although Brand has many attribute values along each branch, only a subset apply, because each Brand only applies to one or two market segments. As a result, CDT never branches by all of the values in Brand, and only branch by a small subset of Brand.

It is possible for the CDT calculation to move Brand lower in the tree by itself, but in order to improve the performance of the CDT calculation in such a case, it can be helpful to direct the calculation to move the attribute with a large number of values lower in the tree.

There is no direct way in the CDT application to force an attribute to be lower, but here are some indirect strategies to use:

- In the Calculation Stage, set the Market Segment attribute to be a Top Attribute. This forces the Market Segment to be at the top of the tree, and so Brand will not be at the top of the tree. This can improve the performance of the Calculation Stage of the CDT application because it has fewer options to consider when expanding the tree.
- Set multiple attributes as the Top Attribute. It is possible that multiple attributes in combination delineate the market segment, for example. There could be a main segment and a sub-segment, as two separate attributes. In such a case, set both of them as Top Attribute. The CDT Calculation Stage will still determine the ordering of the Top Attributes.

For more information, see [Functional-Fit Attributes](#).

Grouped vs. Raw Attribute Values

CDT supports grouped attributes, which turns the raw attribute into one with many fewer values by grouping the attribute values into a small number of bins (for example, 3 to 5). CDT does not have an automated way of performing this grouping, so it is best if the grouping is done in consultation with the retailer, who may have specific requirements.

For example, a retailer may want the following grouping of soft-drink brands: {Soda A, Soda B, everything else}, because they are most interested in their two main brands and are willing to bin all other smaller brands together. The grouped-attribute approach is primarily useful for retailers who are not concerned with every specific value of an attribute that may have a large number of values.

A CDT with a grouped attribute will have branches only for the groups, and not for the original raw values within the groups. In the example above, the branches for Brand would consist only of Soda A, Soda B, Everything Else. This approach can provide additional insights into shopping behavior that would not be available with the raw-attribute approach. For example, if the Brands were grouped into three groups, each representing a particular price tier (say High, MainStream, and Budget), then the CDT can show the behavior of Mainstream customers vs. the behavior of Budget customers. The portion of the CDT underneath Mainstream would look different from the portion of the CDT underneath Budget. In essence, the portions are CDTs that show how each type of customer shops. This insight would not be available using only the raw Brand attribute.

Attribute Splitting

Another approach to handling attributes that have a large number of attribute values is to break them into two attributes (known as attribute splitting). For example, if the Color attribute has many values representing combinations of colors, it may be best to break the single color attribute into two attributes, with one attribute representing the primary color and the second attribute representing a modifier. However, this is an advanced technique, and grouping is recommended over attribute splitting.

Functional-Fit Attributes

A functional fit attribute is one where there is no substitution across the attribute's values. For example, batteries of different sizes cannot be substituted for one another. Any category where size determines the functional suitability of the item will have size as a functional-fit attribute. Information about which attributes are functional fit ones must be loaded into the CDT application.

Designating an attribute as functional fit can also be useful any time the attribute is unlikely to have substitution across it (for example, caffeinated vs. decaffeinated coffee). This is not exactly functional fit; however, substitution is unlikely, so it is better to mark the attribute as functional fit.

Functional-fit attributes always appear at the top of the CDT. The order of the functional-fit attributes will be some arbitrary order, but the order is not meaningful since there is no sense in which one functional fit attribute is more important than another. What the functional-fit attributes do is partition the set of items in a category into their own separate groups, each of which then has its own CDT.

This same effect can be achieved via the UI in the Calculation Stage, by using the Top Attribute functionality of the pop-up called Category Attributes Setup. Using the Top Attribute check box in this pop-up indicates to the Calculation Stage that the attribute should be put at the top of the tree.

CDT Hierarchy Used in the AIF Alternate Hierarchy

The attribute value branches from the CDT application can be extracted and used as a basis for generating alternate hierarchies. The AIF alternate hierarchy building flow has a mix of manual steps (updating some STG tables from the UI to specify the levels of the alternate hierarchy being built and the source tables/columns from which to obtain the alternate hierarchy data) and automated steps/jobs (to build the full hierarchy and populate the various RSP hierarchy tables using the information from the manual step). The information is only used within AIF Apps, and given that, the hierarchy data will not be sent back to AIF Data.

To enable this feature, the RSE parameter `CDT_CREATE_ATTR_ALT_HIER_IND` must be enabled in the Manage System Configuration screen. When you execute the AdHoc batch `CDT_CM_ATTR_BRANCH_EXTARCT_ADHOC_PROCESS`, this extracts the attribute branches for approved CDTs and loads them into the staging tables.

After the job completes and the data is staged, the alternate hierarchies will be generated once the existing nightly alternate hierarchy batch executes. Note that the new alternate hierarchy is based on the extracted, approved CDTs only.

When running the job, if the RSE parameter `CDT_CREATE_ATTR_ALT_HIER_IND` is set to **Y**, then a new hierarchy type will be inserted in the alternate hierarchy staging tables. The name of the alternate hierarchy will have a name pattern of `CDT_PROD_ATTR_HIER%`. This name must not be modified, as the alternate hierarchy process will specifically look for this pattern.

The number of levels generated for the new hierarchy depends on the value of `CDT_ATTR_BRANCH_EXTRACT_LEVEL`. By default, Level 1 is Company, Level 2 is the Category and lowest level is the SKU. Anything between the Category level and SKU level is determined by this parameter. With this, a value of three in `CDT_ATTR_BRANCH_EXTRACT_LEVEL` will have six levels of hierarchy, with levels 3 to 5 being the attribute value levels.

The `CDT_CREATE_ATTR_ALT_HIER_IND` flag is reset to **N** after end of the AdHoc job. The nightly batch for alternate hierarchies executes and the new hierarchies are generated and stored in the `RSE_PROD_HIER` tables.

If you want to create a new hierarchy based on new set of approved CDTs, then you must re-enable the indicator `CDT_CREATE_ATTR_ALT_HIER_IND`, and repeat the above steps.

Note that old hierarchy types generated are kept in the system and are not deleted. You must follow the removal of alternate hierarchy type process to clean up types that are no longer required or are not in use.

Customer Segments

The CDT application can calculate CDTs by customer segment. Customer segments are set of groupings of the customer IDs that are used to identify the transactions. The retailer must provide the customer segment information as a data feed.

Customer segments are useful for retailers who believe that different customer segments shop differently. They may want CDTs by segment only for particular categories. The Calculation stage provides check boxes that control whether or not the run generates CDTs by segment. You can use these check boxes to calculate CDTs by segment for particular categories only.

Frequently, the groups will overlap, since a customer can belong to more than one segment. The CDT application functions normally in this case, since a separate CDT calculation is done per segment.

Location Hierarchy

The CDT application supports calculating CDTs by location hierarchy. The lowest level of the hierarchy should be above store; in general, calculating CDTs per store is not recommended. Per-store CDTs may have too little data to be reliable, and the calculation time involved can be quite long.

Some retailers may have stores that vastly differ in size and assortments. For example, a grocery chain may have both convenience stores and supermarkets. It may be necessary to arrange a separate calculation of CDTs for convenience stores vs. supermarkets, because people may shop differently at the two types of stores and the assortments may be different at the two types of stores.

One approach to this is to arrange a separate calculation by creating separate store clusters for convenience stores vs. supermarkets. The CDT application has the capability of calculating CDTs for each element of the location hierarchy, so it can calculate CDTs for the separate store clusters and thus produce separate CDTs for convenience stores vs. supermarkets.

Setting Up Categories

In general, a category is a set of items that are substitutable with each other (if there are no functional-fit attributes). The categories at a retailer can all be derived by choosing the correct level of the merchandise hierarchy at the retailer. The CDT application configuration supports choosing which level of the merchandise hierarchy is to be used as the category level.

A retailer may want categories that consist of unions of nodes of its merchandise hierarchy, because no level of its merchandise hierarchy suffices as the category level. The CDT application does support this, in that it allows defining an alternate merchandise hierarchy, where the categories can consist of arbitrary collections of items. However, before investing time in setting up an alternate hierarchy, make sure that it is necessary for meaningful CDT calculations.

Consider a category that consists of related though not substitutable products. For example, a single category of hair products can include both shampoo, conditioner, and items that are both shampoo and conditioner. There may be other hair-care related products in the category as well. A reasonable approach is to create an attribute called "Item Type" or "Market Segment" to indicate why the customer is buying it. The Market Segment attribute will segment the category into several sub-categories, and in the Market Segment, attribute should be set as a Top Attribute (see [Setting the Top Attribute](#)).

Calculating Customer Decision Trees

This section suggests ways to using the stages of the CDT application effectively.

Setting the Top Attribute

The Category Attribute Setup pop-up in the Calculation Stage contains check boxes that force particular attributes to the top of the tree. This is useful in several cases:

- The category has an attribute that has a large number of values (50 or more). See [Position of the Attribute Within the Tree](#).
- The category has a functional fit attribute. See [Functional-Fit Attributes](#).
- The category has an attribute that distinguishes market segments or product use. See [Setting Up Categories](#).

- You want to use the same top attribute that is present in a CDT from another source in order to make comparisons with that CDT easier.

It is possible to set more than one top attribute by checking multiple check boxes. In this case, all of the attributes will be at the top, but the Calculation Stage will determine the ordering of the attributes along each branch. This is useful if, for example, there are several attributes that together determine market segment.

In the case of using the top attribute as a market-segmenting attribute, it is possible to experiment with not using this attribute as the top attribute and letting the Calculation Stage determine the attribute ordering. This is useful if the market-segmenting attribute is not truly a functional fit attribute; that is, consumers can substitute across some of the market segments. For example, in the Cookie category, most likely customers can substitute across most of the market segments, because almost all cookies are desserts. In such a case, the Calculation Stage can give additional insight, by showing, for example, that customers actually shop by brand, so that even when they substitute across market segments they tend to stay with the same brand. This can be valuable information. However, if the retailer is interested only in substitutions within market segments, then it is proper to designate the market segment attribute as the Top Attribute.

However, in the case where the category has a very large number of items (greater than 1000), or the category has an attribute with a large number of values (50 or more), it is unwise to try such experiments, because the Calculation Stage may run too long. For such categories, setting the market-segmenting attribute as the top attribute is the best approach.

Excluding Attributes from the Calculation

The Category Attribute Setup pop-up in the Calculation Stage allows excluding particular attributes from the calculation. Use this to avoid meaningless attributes in the tree and also to decrease the calculation time of the Calculation Stage. Include only attributes in the tree that actually affect customers' purchasing. For example, Country of Origin may or may not be a relevant attribute, depending on whether it is visible on the package and plays a role in customers' decisions. Excluding such attributes will not only create a more useful CDT, but will also cut down on the execution time of the Calculation Stage.

Handling of the Brand Attribute

Almost all categories will have a Brand attribute. The power of brands is well-known in retail, and in most categories, customers tend to stick with the same brand. Because of this, the Brand attribute will tend to show up near the top of the CDT. This is the correct scientific result, but not necessarily a useful one, for two reasons:

- It is known that customers shop by brand.
- Brand may have many attribute values, and the resulting tree will be shallow if Brand is high in the tree (see [Position of the Attribute Within the Tree](#)).

Here are some strategies for getting around these effects:

- Exclude Brand from the tree (see [Excluding Attributes from the Calculation](#)). The resulting tree will describe customer behavior in the other attributes. This indicates customer behavior, assuming that they shop by Brand. Given that they shop by brand, what are the effects of the other attributes on their purchasing behavior? CDT answers that question.
- Use the Top Attribute functionality to move Brand lower in the tree. See [Setting the Top Attribute](#).
- Group the brands, so that Brand becomes a grouped attribute. See [Grouped vs. Raw Attribute Values](#). This is a reasonable approach if taken in conjunction with the retailer, and

can offer additional insight into shopping behavior not available without grouping. However, this approach is best taken as a phase 2 task, rather than immediately.

Limitations of the CDT Calculation

Because the CDT calculation uses historical data, the resulting CDT depends on the historical assortment represented in the data. If a particular attribute value does not have any representation in historical assortments among a particular group of stores, then the CDT for those stores will not have this attribute value in it. Similarly, if the assortments carried many more items of a particular attribute value compared to another attribute value, which limits the customer's choices, this can affect the CDT.

It is important to select historical data that reflects the retailer's current assortment, if the retailer has changed assortments in the last few years.

Choosing the Time Interval

The data used to calculate CDTs can be restricted to specific time intervals in the Data Setup stage. Thus, it is not necessary to use all of the available historical data to calculate CDTs. Some possible reasons for restricting the data to specific time intervals are:

The retailer may decide that particular time intervals, such as the two months before Christmas, represent periods where the buying behavior of its customers is significantly different for certain categories. In this case, you can run the CDT application for just for these categories. Choose these categories in the UI, and then also choose the particular time intervals where the retailer believes shopping behavior is different.

If the retailer has changed business practices for certain categories, it is better to exclude the historical data from before the change, so that the CDTs reflect the retailer's current business practices and assortments, not the past ones.

One caution about selecting time intervals: there is always the danger of selecting too narrow a time interval, so that the amount of historical data in the interval is too little. See [Transactions Data Requirements](#).

In general, it is better not to restrict the data too much.

Understanding the Filter Settings

The Data Filtering stage of the CDT UI can be used to filter the data in order to remove outlier data that may result in incorrect CDTs. The user can adjust the values for the filters in order to control the extent of the filtering. The Data Filtering stage has five filters.

The three absolute filters have values that represent absolute limits that the data in question must pass in order not to be filtered out. For example, the maximum on missing attribute values states an absolute maximum that items must meet in order to be used in the CDT calculation. Items that have more than the maximum allowable missing attribute values will not be used in the CDT calculation.

The two relative filters have field values that are relative to the median of each category. The filters use median instead of the more-common average because the median is less likely to be affected by extreme outliers in the data. The average can be brought up (or down) by a single extreme outlier; this cannot happen with the median.

For example, the Transaction History Minimum is a percentage of the median transaction history length for a particular category. It is possible that the transaction history length varies by category. In generating a CDT for a particular category, the goal is to filter out customers who have transaction histories that are too short.

The default value of the filter for Transaction History Minimum is set to 1%, which filters out the customers that are truly outliers for the category because their history length is much smaller than median.

Segments vs. Location

Calculating CDTs by both segment and location hierarchy is not recommended. This calculation generates a large number of CDTs, since it will generate one CDT for each combination of location and segment, which takes a large amount of computation time. The large number of CDTs generated are not considered practically useful. You should either generate CDTs by segment, using Location Chain, or generate CDTs by location, using Segment Chain.

For a first calculation of CDTs, it is best to calculate them at Segment Chain/Location Chain. This quickly generates one CDT per category. It is a good way to check that everything has been done correctly and that the CDTs being produced are not unreasonable.

Setting the Escalation Path

The last stage in the CDT application involves setting the escalation path. If you are using only the segment hierarchy or only the location hierarchy, the escalation path is simply the hierarchy that you are using, and you set the escalation path according to the hierarchy. If you are using both a location hierarchy and a segment hierarchy, then usually you should set the escalation path to go up the segment hierarchy first, and then the location hierarchy. It is better to use only one of the hierarchies.

When using both hierarchies, the escalation path is necessary in order to tell the application which parent it should go to when moving up from a given segment/location node. With both hierarchies in play, every segment/location node has multiple higher-level nodes that do not lie along a single path. The escalation path is necessary to tell the application in what order the higher-level nodes should be considered. When only one hierarchy is used, the higher-level nodes form a single path.

How the CDT Score is Calculated

The terminal nodes of a CDT are the lowest-level nodes in the tree. The terminal nodes of the tree partition the items in the category. The items within each terminal node should be quite similar to each other, and less similar to the items in the other terminal nodes. The terminal nodes provide a clustering of the items in the category. A numerical score for the clustering given by the terminal nodes can be calculated.

Unconstrained clustering using any of the standard clustering algorithms using the similarities as the distance measure can also be created. This clustering can be compared with the clustering score for the clustering by terminal nodes. The terminal-node cluster score will be lower than the score for the unconstrained clustering because the unconstrained clustering had no constraints when performing the clustering. The closer the terminal-node clustering score is to the unconstrained score, the better the CDT. The CDT score in the CDT application is represented as a percentage of the unconstrained clustering score.

Typically, you should eliminate any CDT that has a score of below 60 percent, using the Pruning stage of the application.

Understanding CDT Pruning

The Evaluation Stage of the CDT application performs an operation called pruning, in which entire CDTs are removed. In the Evaluation Stage, the CDT as a whole is deemed reliable or

not. An unreliable CDT is removed in its entirety; there is no automatic mechanism for making small adjustments to a CDT. The only mechanism the CDT application has for making small adjustments to a CDT is the manual editing of a CDT allowed in the CDT editor.

Overriding the CDT Calculation

It may be necessary, because of prior knowledge concerning the business of the retailer, or knowledge about the historical transactions at the retailer, to override portions of the calculation performed in the Calculation Stage of the CDT application. The override mechanism there allows you to specify what the topmost attributes of the CDT should be. For example, from an understanding of the retailer's business, it may be clear that in a particular category, brand should be at the top level of the tree. The override mechanism allows you to specify brand as the top level of the tree. The override mechanism is also flexible enough to allow specifying only the top level of the tree, while the rest of the tree is filled in by the usual calculation.

While it is possible to obtain the same effect by manual editing of the CDT, manual editing is much slower, especially if you have generated multiple CDTs for each category.

Using the Calculation Stage

This section provides step-by-step instructions for setting up the Calculation Stage, with a few comments on using the other stages. The focus here is mainly on the Calculation Stage, because the settings in this stage can directly affect how the CDTs look and because the Calculation Stage generally takes up most of the execution time.

If you are just beginning to use the CDT application, experiment with smaller categories (fewer than 1,000 items) initially. Smaller categories are easier to work with because they take less execution time in the Calculation Stage than larger ones, so it is easier to do multiple runs and examine results.

Setup Stage

When first starting to use the CDT application, it is best to set up only one category at a time in the Setup Stage. In this way, each run is for one single category. It requires some experience to include multiple categories in the same run, and it is not recommended as a starting point. The instructions assume only one category has been set up in this run.

Before selecting a category for a CDT run, review the data requirements in [Transactions Data Requirements](#) to be sure that your desired category meets the data requirements.

Data Filtering Stage

This stage is usually straightforward, in that the default values of the fields are usually suitable. However, it is important to check the Data Filtering Summary at the bottom of the screen *after* the stage has completed running. You must click **Refresh** in the summary table in order to see the results related to the latest run. Check each filter in the summary to see how much data it filtered out. If too much data was filter out, then determine whether the data may have a problem, or whether you need to adjust the filter so that it is less stringent.

If this is the first time you have run the Data Filtering Stage on a particular category, then you should run only the Setup Stage and the Data Filtering stage on the category, without running the Calculation Stage. This allows you to check the Data Filtering Summary before spending time running the Calculation Stage. Once you have run the Data Filtering Stage on a category and have checked the Data Filtering Summary, then you can re-run the Data Filtering Stage on

the same category without checking the Summary, unless you have loaded new data for the category.

Calculation Stage

The steps here are simplified to help you get started in properly using the Calculation Stage. Once you become familiar with using this process, you can alter and expand them to use more of the capabilities of the CDT application. The process presented here represents the minimal set of steps needed to produce CDTs and to get you going in the right direction.

Take care during this process so that the Calculation Stage can complete within 1 or 2 hours for categories that have more than 1,000 items or that have an attribute with more than 50 values. The steps detail any additional consideration needed for large categories. After performing a run with these steps, if the time for the Calculation Stage to run turns out to be acceptable, then these restrictions can be relaxed on subsequent runs of the category.

Each step may reference sections of this chapter that can provide further details.

1. Check both top level check boxes (one for Segments and one for Location). With these settings, the Calculation Stage will generate only one CDT, representing the CDT for all customers and all locations. This is the recommended way to start using the Calculation Stage. In particular, these settings are recommended for very large categories, where the calculation time for multiple CDTs may be quite prohibitive and not worth the investment until you have generated one CDT.
2. Exclude any unnecessary attributes (see [Excluding Attributes from the Calculation](#)). It is good practice to exclude unnecessary attributes, but it is even more so when working with large categories in order to avoid unnecessary computation. For large categories, consider excluding attributes that you know are less important to the retailer, even if they may have an effect on customers' purchasing in the category.
3. Handle Brand properly. Brand frequently has many attribute values, and handling it properly is especially important when the category also has a large number of items (1,000 or more). You can skip this step if the category has fewer than 1,000 items. See [Handling of the Brand Attribute](#).
4. Set Top Attributes properly. In particular, if the category has some type of market segment, product type, or product usage attributes, then force these attributes to be at the top of the tree. If the category has a large number of items, then it is likely to require some of these attributes, because with that many items, the items will likely have different segments or types. It is unlikely that the entire set of items is completely interchangeable in the customers' mind, and so it is proper to put segmenting attributes at the top of the tree. In addition to being scientifically proper, this decreases the execution time of the Calculation Stage because there are fewer combinations for the stage to consider and because other attributes such as Brand with a very large number of values will be moved down the tree, where fewer items are involved in the calculation. For more information, see [Setting the Top Attribute](#).
5. For large categories, consider setting the SKU Percentage of the termination condition to a lower value (possibly 0%). A value of X in this field in the Calculation Stage UI specifies that a branch will end when a node on the branch contains fewer than X% of the items in the category. If X = 5%, which is the default, and the category contains 2,000 items, then the branch ends when a node on the branch contains fewer than 100 items. A threshold of 100 is probably too high, and if left at 5%, various branches may not be expanded to their full extent. If the value is expanded until a node on the branch contains fewer than 10 items, the SKU percentage field must be set to $10 / 2,000 = 0.5\%$. However, the field only accepts integer percentages, and so it must be set to 0%, which will let the Calculation Stage use other internal criteria to end a branch. This field can also be used in a reverse manner; that is, by setting a higher value the tree will become shallower and the

calculation time will be reduced. For trial runs, you may wish to leave it at 5%, and see how far the branches are expanded before trying a run with a setting of 0%. For smaller categories, with fewer than 1,000 items, using the default of 5% is likely to be reasonable and no adjustment is needed. The use of percentage in this field can also be handy if you are performing runs that have more than one category. If the categories are related, so that you want trees of roughly the same depth for them, the percentage nature of the field will help produce this result.

Advanced Use

The process described in [Calculation Stage](#) is intended as a starting point, and is the shortest path to getting one CDT per category. Once that has been achieved, it is possible to consider some more advanced uses of the CDT application. Here are some suggestions for these more advanced uses. These are not steps to be performed but individual suggestions.

- Generate CDTs that are specific to location or segment. The Calculation Stage can generate CDTs that are segment specific or location specific, by unchecking the appropriate Top level check boxes in the stage. This makes it possible to see whether purchasing behavior differs by segment or by location. Note that on large categories with more than 1,000 items, such a calculation can take many hours, because the stage must calculate one CDT per segment or per location. Note also that it is *not* advisable to uncheck both check boxes, as that will produce a CDT for each segment/location combination. This is a large number of CDTs that will take a very long time to run.
- Set up grouped attributes. See [Grouped vs. Raw Attribute Values](#).
- Experiment with setting different attributes as the top attribute, or with not setting a top attribute at all. See [Setting the Top Attribute](#). Different settings here can produce different insights. However, keep in mind the points raised in [Handling of the Brand Attribute](#).

4

Demand Transference

This chapter provides details about the use of the Demand Transference application.

DT and CDT

The DT and CDT applications differ in significant ways. The CDT application has more stringent requirements for data than the DT application. CDT requires customer-linked, frequent transactions. Many retailers in various areas of retail do not have this type of data readily available. DT only requires SKU-store-week sales-units aggregates.

Demand Transference Model

A mathematical model of how the transference happens is required in order to calculate the transfer of demand in response to assortment changes. It is essential to understand the model at a basic level in order to best use DT. DT generates parameters that go into the model, so an understanding of the model can help when using the DT parameters.

The model is known as a cannibalization model. In this type of model, each item in an assortment has an associated value called its "full demand," which is the demand the item would have if it were the only item in the assortment. The full demand of an item is then multiplied by a factor, called the "cannibalization factor," which has a value of 1 if there are no other items in the assortment, but becomes progressively less than 1 as more and more items are added to the assortment. As the assortment becomes larger, the demand for each of the items decreases from its full demand because of cannibalization. The reverse is also true. If items are removed from the assortment, then the cannibalization factors increase, representing demand transferred from the removed item to the items remaining in the assortment. The cannibalization factors decrease from a value of 1 when the assortment becomes larger, and increase (up to a limit of 1) when the assortment becomes smaller.

The degree of change in an item's cannibalization factor indicates how similar the added items are. Item A's cannibalization factor will decrease more for added items that are very similar to A. The similarity of items is a key input to the Demand Transference model.

The cannibalization factor of an item accounts for similarities and also for a quantity called "assortment elasticity." The assortment elasticity determines how much of a decrease in the cannibalization factor occurs due to the addition of items of a particular similarity. The assortment elasticity is a number that depends on the particular category for which demand transference is being calculated. In one category, adding item B to the assortment may cause item A's cannibalization factor to go from 0.7 to 0.6, whereas in another category, adding an item Y may cause item X's cannibalization factor to go from 0.7 to 0.5, even though the similarity of X and Y is the same as the similarity of A and B. In other words, similarities alone are not enough to calculate cannibalization factors. The assortment elasticity is necessary to tell us, for each category, how much change in cannibalization factors will occur for items of a given similarity.

The two components of the cannibalization factor, the similarities and the assortment elasticity, are calculated by DT from historical data. (The similarities can also be imported instead of calculated.) DT then exports the similarities and the assortment elasticities to any applications that want to calculate demand transference. It is up to the consuming application to properly

use the cannibalization model in conjunction with the exported similarities and assortment elasticities to calculate transfers of demand when assortments change.

Note that demand transference only occurs within a category. All calculations are based on items cannibalizing each other, and there are no complimentary (halo) effects. DT calculates assortment elasticities at a level always higher than item. A single category/segment/location combination receives just one assortment elasticity.

An Example

This simple example explains how applications such as Category Management Planning and Optimization (CMPO) use the demand transference model to generate forecasts after assortment changes. In an assortment in the Cookies subcategory, one cookie SKU is removed from the assortment. The cannibalization factors of the rest of the items increase, because each is now cannibalized less after the removal of the one SKU. Because the cannibalization factors increase, the model predicts an increase in sales in accordance with the increase in the cannibalization factors. The removal of the SKU caused these increases, and some of the SKU's demand has transferred to the other SKUs.

Historical Similarity Data

DT has two different options for obtaining similarities. It can calculate them by itself or it can import them from CDT.

If the retailer has not implemented CDT for a category, then of course only the first option is possible.

The second option is recommended for a category only if the retailer has implemented CDT and run it for the category, since only in that case are similarities for the category available from CDT.

If CDT similarities are available, the recommendation is that you use them, instead of having the Similarity Calculation stage calculate its own similarities. The similarities from CDT are generally preferable to the attribute-based similarities that DT can calculate on its own because CDT similarities do not rely on attributes. They are extracted purely from historical transactions data.

The transactions data held in the RADM schema is used to feed both CDT and to generate the SKU-store-week aggregates for DT, so in option 2, consistency between the similarities and the SKU-store-week aggregates is automatic.

The similarities obtained from CDT may not cover all of the SKUs that are currently in the historical data loaded for DT. For example, it is possible that since the CDT similarities were calculated, the retailer has added some new SKUs to some assortments. This situation requires no special handling, because DT can extend the CDT similarities to cover the added SKUs. This extension does require attribute values for the new SKUs.

Note that the CDT similarities for a category may be only at Segment-Chain/Location-Chain. In this case, there is only one set of similarities for the category, or they may exist at various levels of the location hierarchy or the segment hierarchy, depending on what options the user selected in the CDT application's Calculation stage.

Historical Sales Data

DT requires SKU-store-week sales-units aggregates. The data loader for DT automatically produces the needed SKU-store-week sales-units aggregates from transactions data that is

held in the RADM schema, so it is not necessary to implement a separate loader for SKU-store-week aggregates.

Typically, the data cannot be aggregated to a higher location level than store because different stores usually have different assortments. Some atypical cases can occur in which aggregation across some stores is legitimate because the assortments are the same or nearly so, but this is generally not the case.

In addition to the SKU-store-week sales-units aggregates, DT also requires promotion data. A flag indicates which SKU-store-weeks contain major promotions (ones that caused a very large increase in sales units, such as three times normal). DT uses the promotion data to flatten the promotional spikes in the SKU-store-week sales-units aggregates. DT uses the flattened data called the baseline to calculate assortment elasticity.

DT calculates the assortment elasticity by examining the historical assortment changes and seeing their effect on base sales rates of the items remaining in the assortment. Promotional spikes can affect this calculation by obscuring the true effect on base sales rates. These promotional spikes are removed to decrease the sales rates back to their base rates.

For example, suppose the historical data for a store S indicates that the Cookies assortment has one fewer SKU in week 10 compared to week 1. That is, a cookie SKU was removed. To see where the demand from this SKU transferred to in week 10, the sales units of the remaining SKUs between week 1 and week 10 are compared. This comparison is made across many pairs of weeks (though not all possible pairs of weeks). A promotion in week 10 of particular items can interfere with the analysis of the changes in demand that were due to the assortment changes.

Note that, in CDT, the effect of promotions is left in because it is an external influence that helps cause switching behavior in customers. However, in DT, promotions can affect the calculation of demand transference in the case where one item in the category is promoted and another is not, which is why the promotions are flattened.

An alternative to flattening the promotions is to ignore the SKU-store-weeks where promotions occurred. However, to implement this, it is necessary to ignore all SKU-store-weeks in any week where a SKU-store was promoted, because it is not clear what the effect is on the other items when one SKU is promoted and the others are not. Removing that many SKU-store-weeks can leave little data remaining, especially since many retailers promote quite frequently. For this reason, it is better to flatten promotions and keep more data instead. Similarly to the way in which promotions are handled, the calculation of the baseline also involves removing short-term downward spikes that are due to very short-term stock outs. Note that inventory information is not an input to the DT system, and so the algorithm finds large but short-lasting dips in weekly sales units and fills those in. This handling of out-of-stock is not related to long-term out-of-stock conditions, discussed in a later section.

The Role of Attributes in Calculating Similarities

Without customer-linked transactions data, DT must use the attribute values of the SKUs to calculate similarities. The similarity of two SKUs is based in part on how many attribute values they have in common (the more in common, the higher the similarity of the two SKUs). The attributes used in the calculation are the raw attributes, not the grouped attributes that CDT uses. So it is not necessary to group the attribute values for DT.

Because the attributes play such an important role in calculating similarities, attribute quality is important when DT performs the calculation.

Attributes are also used in performing any necessary extensions of the CDT similarities to cover new SKUs.

Note that the similarities calculated by DT are only at Segment-Chain/Location-Chain. In contrast, the CDT similarities can be at multiple levels.

Attribute Data Requirements

The attribute values for the DT calculation must meet the following requirements:

Set of Attributes

Each category is characterized by a unique set of attributes. These attributes differ from category to category. For example, for yogurt, the attributes might be size, flavor, brand, fat percentage, and pack size. For chocolate, the attributes might be size, brand, milk/dark, nut type, and package type. Two categories can both have brand, but the brand attribute will have different values for each of the categories. So brand is actually a different attribute for each category.

Mapping

Each item in the category must be mapped to its set of attribute values. This information must be obtained from the retailer. Null values are acceptable as long as they are not too numerous. DT can still run even if some attribute values are listed as null for some items in a category, but too many null values decrease the reliability of the generated DTs. In particular, too many SKU pairs may come out as less similar than they should be, which would decrease demand transference between those pairs (which leads to an underestimate of demand transference in applications such as CMPO).

Null values have a particular use in accommodating categories that are actually a union of more than one category. See "[Setting Up Categories](#)" for more information.

Significance

The attributes for a category must be the ones that the customers actually pay attention to when shopping in the category. They are attributes that actually affect the customers' purchasing decisions.

Note that the similarity calculation will still complete even with attributes that do not affect customer behavior, but the similarities produced will be less distinguishing. For example, a category has a Supplier attribute, which indicates for a given product which supplier shipped an item to the grocer. This attribute may be important to the grocer for accurate bookkeeping, but it has no effect on the customer's purchasing behavior because it is not reflected in the item itself nor is it something that the customer is concerned about. However, if it is included when setting up attributes, then the effect would be to increase the similarity of items that were from the same supplier. This is a false similarity, since it does not reflect how the customer actually views these items. In particular, if the supplier is a duplicate of the Brand attribute, then the similarity of products within the same Brand would be unintentionally increased.

The process of obtaining attributes for a category and performing a mapping of items in the category to attribute values is likely to require a significant amount of time and labor, even if the retailer has the information available, since this must be done for every category.

Guidelines on Number of Attributes and Attribute Values

The number of attributes and attribute values must be enough to distinguish the SKUs within a category. That is, for a given set of attribute values, the number of SKUs in the category all having those values must be a small number. A maximum of seven SKUs is recommended. For example, the Cookie category at a grocer has only three attributes, Brand, Package Size, and Organic. If Brand has seven values, Package Size has three values, and Organic has two

values (either Yes or No), then the total number of combinations of attribute values is $7 \times 3 \times 2 = 42$. For 600 different cookie SKUs, the average for each combination of attribute values will represent $600 / 42 =$ approximately 14 different SKUs. The distribution of SKUs among the 42 different sets of values will not be an even 14, as some sets of values will have much more than 14, while others will have less. The three attributes alone are not enough to provide enough distinguishing power among the cookie SKUs. If Flavor is an important determinant of customer purchases, it should be added to the Cookie category. The guideline of a maximum of seven indicates that additional attributes are necessary. It is worth examining those sets of attribute values that have the largest number of SKUs associated with them in order to see what attributes can be added to reduce the number of SKUs. It is not just the number of attributes that is important, but how many values each attribute has. For example, if Brand had 100 values instead of seven values, then the total number of attribute-value combinations is $100 \times 3 \times 2 = 600$. It might seem that an easy way to achieve the maximum of seven is to expand the number of values in the attributes. However, this results in each SKU being similar to only a small number of other SKUs. For a single attribute for cookies with 600 different values, it might then be possible to assign one value to each cookie SKU, separating all of the 600 cookies SKUs with a single attribute. However, this would make each cookie SKU completely dissimilar (similarity of 0) from all other cookie SKUs, and the result would be no transference between the SKUs. Putting all 600 SKUs each into a separate Brand causes a complete loss of any similarity information among the SKUs, and no transference will result. For an opposite example, consider 11 attributes, each with only two values. There are a total of $2^{11} = 2048$ combinations of values, so that may be enough to encode 2,000 SKUs, even though there are only $2 \times 11 = 22$ distinct attribute values over the 11 attributes. In general, having more attributes is better, and it is better to increase the number of attributes rather than increase the number of attribute values of a single attribute. However, this is not always possible, and it is better to have the attribute with many values than not have the attribute at all. Flavor, for example, can have many values, as can Color. See "[Avoiding Attributes with Many Values](#)" regarding attributes that have many values. The more SKUs in the category, the more attributes and attribute values will be needed to achieve the maximum of 7.

The Effect on Similarity Values

Suppose the set A of SKUs consists of 22 SKUs, all with the same attribute values, and the set B of SKUs consists of 25 SKUs, all with the same attribute values (but a different set of attribute values from set A). If the set A consists of cookie SKUs all with a package size of Small, and set B consists of cookie SKUs with the same attribute values as A except the size is Medium, then every SKU in A has a similarity of 1 to every other SKU in A, and every SKUs in A is similar to at least 22 other SKUs. Every SKU in A is similar to every SKU in B, since they only differ in one attribute value (namely size). So a SKU in set A is similar to at least $21 + 25 = 46$ SKUs, which means that if a SKU in A were deleted from an assortment, its demand would have significant transference to about 46 other SKUs, assuming all 46 remained in the assortment. It is possible that a SKU in A being similar to 46 other SKUs in fact represents reality, but if it does not, then using additional attributes that distinguish the SKUs in A and in B will reduce the number of similar SKUs.

Avoiding Attributes with Many Values

Attributes with a large number of values occur frequently. For example, a color attribute in any clothing category might have several shades of each color. Midnight blue, sea blue, and sky blue may all be separate attribute values of the Color attribute; the problem is that in the similarity calculation, a midnight blue item and a sea blue item would be considered completely dissimilar colors, because the two color attribute values are different; in reality, because they are both shades of blue, they should be somewhat similar. One solution is to split the color attribute into two separate attributes, a primary color attribute and a modifier. In this example, the primary color would be blue and the modifiers midnight, sea, and sky.

Functional-Fit Attributes

A functional fit attribute is one where there is no substitution across the attribute's values. For example, batteries of different sizes cannot be substituted for one another. Any category where size determines the functional suitability of the item will have size as a functional-fit attribute.

Information about which attributes are functional fit ones must be loaded into DT. The information is used to perform the similarity extension process of CDT similarities and to correctly calculate attribute-based similarities.

In either case, the functional-fit attributes are used to set the similarity of two SKUs to be 0 if the SKUs differ in any functional-fit attribute. Without the functional-fit information, the two SKUs may have non-zero similarity, and there would be erroneous demand transference between the two SKUs, such as batteries of different sizes.

Designating an attribute as functional fit can also be useful any time the attribute is unlikely to have substitution across it (for example, caffeinated vs. decaffeinated coffee). This is not exactly functional fit; however, substitution is unlikely, so it is better to mark the attribute as functional fit.

One approach to avoiding having to define large numbers of attributes and attribute values is to use functional-fit attributes. This approach does not help achieve the maximum of 7, but it can help decrease the number of SKUs that are similar to a given SKU. For example, with the sets A and B of cookie SKUs, if size were designated as functional fit, then the similarity between SKUs in A and SKUs in B would become 0. However, that designating size as functional fit does nothing about the 22 SKUs in A that all have a similarity of 1 to each other, since their attribute values are all the same. (Similar comments apply to set B.)

If the attributes and attribute values are insufficient to reach the maximum of 7 SKUs per set of attribute values, functional-fit attributes can be used to decrease the number of SKUs to which transference occurs. This is a second-best approach, and it is better to design a proper set of attributes and attribute values, in order to:

- Achieve the maximum of 7 SKUs
- Provide transference between SKUs that should have transference. Using functional-fit attributes reduces transference, but it may reduce it too much and remove transference from pairs of SKUs that should have transference. For example, in the sets A and B, the similarity between a SKUs in A and a SKU in B becomes 0, which does not reflect reality since the SKUs in A and in B share common attribute values except for size.
- Keep the second-best approach as a last resort, in case time is insufficient for designing a good set of attributes for a category.

Using Null as an Attribute Value

Null is a legitimate attribute value for an item to have; it denotes either of the following:

- The attribute value is unknown. In this case, the item has a value for this attribute, but the value is unknown. Attribute data for a category is rarely perfect, it is entirely possible that attribute values for some items were never recorded or are known to be incorrect. The use of null because of imperfections in the attribute data must only be for those cases where it is truly necessary.
- The attribute does not apply to the item. Not all attributes in a category apply to all items, especially when the category consists of sub-categories (see below). The ideal is to define the items in the category so that every attribute for the category applies to every item, but this may not be possible. This usually occurs when an attribute applies to an item only if

the item has a particular value for another attribute. For example, in the meat snacks category, the attribute grass fed only applies if the item has beef for the product type attribute. If the item is turkey then the grass fed attribute must be set to null for the item. The grass fed attribute must be true or false only for beef products.

Using null is not the ideal way of handling the second case listed above. If possible, it is preferable to split the items in the category into separate categories. For example, this means putting beef products into their own category, so that the grass fed attribute applies only to this new category and can be removed from the non-beef products. This is only reasonable if the beef products are not substitutes for the other products in the meat snacks category (which is probably not true since the customer might reasonably substitute turkey jerky for beef jerky). In the case of meat snacks, the best solution is to use null for the grass fed attribute for non-beef products.

The Effect of Null Attribute Values on Similarity Values

A null attribute value for an item means that the item does not match any other item in that attribute, even if the other item also has null for the same attribute. Null as the value of an attribute makes the item less similar to every other item, including items that also have null for the same attribute. If an item has null for every attribute, then this item has a similarity of 0 to every other item, regardless of what values the other items have for their attributes. A value of null for an attribute is thus treated as special in similarity calculations. It is recommended to keep the use of null to a minimum in order to avoid spuriously indicating that items are dissimilar when they are actually similar.

If the intention is to have null for an attribute value, it is essential to actually use null and not create a string such as "none" to use where null is actually meant. Designating a string such as "none" to use in place of null actually does the opposite of what is described above, because two items with a value of none for the same attribute will now be considered similar in that attribute since they have the same value for the attribute. This is directly opposite to the intention of null, as described above.

Categories Containing Sub-Categories

As discussed in [Setting Up Categories](#) it is possible for a category to consist of items from several categories. As it may be impractical to separate the category into its component categories, it is necessary to create attributes for the category that can approximate the separation. The following process describes how to do this, using " the category Oral Care. This category consists of: toothbrushes, toothpaste, dental floss, and mouth wash; it is really four separate categories whose items have all been classified together into a single category.

1. Create one attribute that can separate the category into its component categories. For example, for Oral Care, create an attribute called Product Type that has four values: "toothbrush," "toothpaste," "floss," and "mouthwash." The values of this separating attribute should describe the function of the product. It should separate the category into groups of items in which the items within each group are unlikely to substitute for items in another group. For example, toothbrushes are unlikely to substitute for toothpaste. Constructing a separating attribute is not always as simple as this, as discussed below.
2. Designate this separating attribute as functional fit. This means that no transference will take place between items that are in the different groups specified by the separating attribute. In this case, there is no transference between the four component groups of Oral Care.
3. Create a single set of attributes suitable for all of the component categories. This typically involves taking the union of the attributes that describe each component category separately. For example, the set of attributes that describe toothpaste are different than the

set of attributes that describe toothbrushes, but in this scheme, just take the union of both sets of attributes. Toothbrushes have an attribute called "Bristle hardness," which does not apply to toothpaste, and likewise toothpaste has a flavor attribute that does not apply to toothbrushes. Nonetheless, take all of the attributes together, including the ones for floss and mouthwash. It is possible that an attribute can apply across several of the component categories. For example, only a single flavor attribute is necessary, instead of having three attributes called Floss-Flavor, Toothpaste-Flavor, and Mouthwash-Flavor. Either approach is correct, but having three separate flavor attributes may require more work to create.

4. Assign null values appropriately. For example, for attributes that apply only to toothpaste, set them to null for any toothbrush. Likewise, for attributes that apply only to toothbrushes, set them to null for any toothpaste. [Table 4-1](#) provides complete details for this example.

In step 1, you may not find a separating attribute. Instead, you may need to use an attribute that separates the items into groups but in which some of the groups may have items that can substitute for each other. For example, consider a category that consists of Shampoo, Shampoo and Conditioner, and Conditioner, where Shampoo and Conditioner are products that have shampoo and conditioner in one bottle. In this case, a separating attribute has three values, Shampoo, Shampoo and Conditioner, and Conditioner. If this attribute is set as functional fit, there is no transference between Shampoo and Shampoo and Conditioner. In this example, the approximation may be good enough, since although substitution can take place between Shampoo and Shampoo and Conditioner, it is probably not large.

Here is an example of the Oral Care category and how to set null for its attributes. Assume a separating attribute for Oral Care as described above, and assume, in addition, the following attributes: Thickness, Length, Contains Alcohol, and Flavor. (In reality there would be more.) [Table 4-1](#) shows which attributes apply to which products. Null indicates that the attribute does not apply to the product, and therefore should be set to null for the product.

Table 4-1 Oral Care Category Example

Product Type	Thickness	Length	Contains Alcohol	Flavor
Floss	Applies	Applies	Null	Applies
Mouthwash	Null	Null	Applies	Applies
Toothpaste	Null	Null	Null	Applies
Toothbrush	Null	Null	Null	Null

Customer Segments

DT can calculate assortment elasticities by customer segment. This involves dividing the customer IDs into groups (the groups do not have to be disjoint). Retailers who want to use segments must, as with CDT, create the necessary groupings of customer IDs. DT uses the segments to produce segment-SKU-store-week aggregates of sales units, instead of just SKU-store-week aggregates. The segment-SKU-store-week aggregates are produced by aggregating transactions data, just as with the SKU-store-week aggregates. The difference is that the aggregation is by segment.

There is always a Segment-Chain for the segment hierarchy, and so there is always a segment that contains all customers. The Segment-Chain level of segment-SKU-store-week aggregates is not necessarily the sum of the lower-level segment-SKU-store-week aggregates, because it is possible that the segments are not disjoint (meaning a customer can belong to more than one segment). The Segment-Chain-level aggregates are produced by a separate aggregation of transactions data instead of by aggregating lower-level aggregates.

Using segments allows DT to calculate separate assortment elasticities for each segment. This means that demand transference can differ by segment.

Note that when using customer segments, references in this document to "SKU-store-week" data should be read as "segment-SKU-store-week" data. For example, the SKU-store-week sales-units aggregates mentioned above become segment-SKU-store-week sales-units aggregates.

Location Hierarchy

DT supports calculating assortment elasticities by location hierarchy. The lowest level of the hierarchy should be above store; in general, assortment elasticities should not be calculated per store. Per-store assortment elasticities may have too little data to be reliable. The calculation time involved can be quite large to handle all stores individually. The calculation of assortment elasticities depends on having assortment changes in the historical data, and the store level may contain too few assortment changes to produce reliable assortment elasticities.

Some retailers may have stores that differ in size and assortments. For example, a grocery chain may have both convenience stores and supermarkets. It may be necessary to arrange a separate calculation of CDTs for convenience stores vs. supermarkets, because people may shop differently at the two types of stores and the assortments may be different at the two types of stores.

One approach to this is to arrange a separate calculation by creating separate store clusters for convenience stores vs. supermarkets. DT has the capability of calculating CDTs for each element of the location hierarchy, so it can calculate CDTs for the separate store clusters and thus produce separate CDTs for convenience stores vs. supermarkets.

Setting Up Categories

In general, a category is a set of items that are substitutable with each other (if there are no functional-fit attributes). The categories at a retailer can all be derived by choosing the correct level of the merchandise hierarchy at the retailer. The DT configuration supports choosing which level of the merchandise hierarchy is to be used as the category level.

Demand transference can only occur within the category, since the categories define the sets of items that cannibalize each other.

A retailer may want categories that consist of unions of nodes of its merchandise hierarchy because no level of its merchandise hierarchy suffices as the category level. DT does support this, in that it allows defining an alternate merchandise hierarchy, where the categories can consist of arbitrary collections of items. However, before investing time in setting up an alternate hierarchy, make sure that it is necessary for meaningful DT calculations.

For example, it is possible that the set of all yogurt SKUs at a retailer is not at any level of the merchandise hierarchy. The retailer may have the category Dairy Products, which is too large because it contains yogurt and milk, and the retailer might have the category Store-brand Yogurt, which is too small because it leaves out the yogurt SKUs that are not store brand. In such a case, it may be necessary to set up an alternate hierarchy so that all the yogurts can be put together in their own set. On the other hand, if a level of the existing merchandise hierarchy contains most of the yogurt SKUs, but not quite all of them, an alternate hierarchy may not be worth the effort.

Frequently, retailers will have categories that are actually unions of categories. For example, a retailer might have a Hair Care category that contains shampoo, conditioner, and hair oil. The retailer may not want to separate out this category into three separate categories of Shampoo, Conditioner, and Hair Oil, if, for example, a single person in the organization is responsible for all three. The problem is that these three types of products do not share a common set of

attributes. The attributes describing Hair Oil are not the same ones needed for describing Shampoo or Conditioner. The types of products may share common attributes, such as Scent, but each type of product also needs its own set of attributes. The solution is to define, in addition to the common attributes, a set of attributes for each product type. If an attribute applies only to Shampoo, and not to Conditioner or Hair Oil, then Conditioner SKUs and Hair Oil SKUs should have Null for the value of that attribute. This is a common use of null attribute values and makes it possible to handle the case of a category that is really the union of smaller sub-categories.

5

Using Demand Transference

This chapter provides details about using Demand Transference.

Seasonality in Historical Sales Data

The DT application assumes that, within a category, all of the items at a store have a common seasonality. This assumption is generally correct for long life cycle categories, where each item does not have a predetermined point of obsolescence or where the point of obsolescence is years from the point of introduction of the item. Examples include most grocery categories and basic clothing items. Electronics items frequently have defined life cycles, generally measured in years.

It is important to address the situation in which different items in the same store have different life cycles and those life cycles are short. In this situation, the store may have items that are at various points in their life cycles and there is no common seasonality. This frequently occurs with fashion merchandise (see "[Implementing DT for Fashion Categories](#)").

Assortment Elasticity

An assortment elasticity of 0 turns all cannibalization factors into constant 1, meaning the assortment has no cannibalization. This is unlikely. However, it does show that a small-magnitude value for assortment elasticity indicates a category where cannibalization is small. Similarly, a high magnitude of assortment elasticity indicates a category where cannibalization is large. It is possible for the magnitude to be too large.

It is also possible for the Calculation stage of DT to produce assortment elasticities that are positive. Such positive values for assortment elasticity are an indication that there is some unidentified problem with the data, because a positive assortment elasticity means cannibalization factors increase with increasing assortment size, which in turn means each item in the assortment sells more the larger the assortment gets. In the Evaluation stage of DT, such positive assortment elasticities are removed and replaced by assortment elasticities from escalation.

Here is an example that explains the cannibalization model. In this example, all the cannibalization factors are equal; in a real example, the factors would all be different.

Identical, or nearly identical, items are being added to an assortment. If only one of these items is added to the assortment, it takes the entire market share for the item. When another item that is extremely similar is added, the two items split the market share evenly between them. The cannibalization factors are now half for both items. The addition of a third such item creates a three-way even split, one-third for each and the cannibalization factors are each one-third. As more such items are added, the cannibalization factors approach but never reach zero. (Parenthetically, this example also shows how adding items to an assortment does not necessarily produce more market share overall for the assortment, since the new item may simply siphon off sales of existing items.)

The cannibalization factor is actually a power-law, that is, the assortment elasticity enters into the cannibalization factor as an exponent. The cannibalization factor consists of a positive value, the Total Assortment Effect (TAE), raised to the assortment elasticity. Each item in an assortment has its own TAE; the TAE increases as items are added to the assortment.

Therefore, the assortment elasticity is a negative number, in order for the cannibalization factor to decrease as TAE increases. (In the example, the TAE could simply be the count of the number of items added so far, and the assortment elasticity would then be -1 , thus producing $1/2, 1/3, \dots$) Conversely, the TAE also decreases as items are removed from the Assortment.

Note the similarity to the more conventional idea of power-law price elasticity, which involves a price raised to a negative power (the negative power being the price elasticity). In the cannibalization model, the TAE plays the role of price. Like price, the TAE can change week to week because the assortment can change week to week and the TAE is designed to reflect the current assortment.

The cannibalization factor also accounts for the similarity of the items being added to the assortment, so that similar items cannibalize each other more than non-similar ones. The similarity values are used to calculate the TAE; higher similarities produce a larger TAE, providing a larger decrease in cannibalization factors.

The TAE also depends on the relative sales rates of the items in the assortment. Although higher similarities increase TAE, as discussed above, that cannot be the whole story, because cannibalization depends on the sales rate of the item doing the cannibalizing. For example, consider two items A and B that have equal similarity to an item C that is already in the assortment. Adding item A will cannibalize item C, and so will adding item B; however, if A is a low selling item and B is a high selling item, adding B will cannibalize C more than adding A will. So the TAE must account for the relative sales rates of the items in the assortment in addition to accounting for similarity. The TAE does this through the sales index of an item, which is the sales of the item divided by the average sales of the items in the assortment (total sales divided by the number of items in the assortment). This calculation is done in each week, with the assortment in that week. The following example demonstrates these concepts together and also shows some of the robustness of this model. Consider an assortment that includes items A and B and in which A has a vastly higher sales rate than B. The TAE for item A includes a term for item B, and as discussed above, the contribution of B to the TAE depends on its sales index. In this case, the sales index of B is quite small, because A is so large. Removing B from the assortment will change the TAE of A very little, and so the change in A's sales due to removal of B is quite small. In a situation where A's sales become increasingly large, the sales index of B will go to 0, and the effect of the removal of B on A tends to 0. This is as it should be, since when A becomes so much larger than B, the removal of B cannot have much effect on the sales of A.

The cannibalization factor depends on both the similarity values and the assortment elasticity. It might seem that similarity alone determines cannibalization, as a similarity of 0.5 between items A and B means that A takes half of B's share if A is added, but that is not the case. In particular, by separating the concepts of TAE and assortment elasticity, the model is more robust; if all of the similarity values are biased lower or higher for some reason, the bias can be accounted for by adjusting the magnitude of the assortment elasticity so that the cannibalization factors are still correct.

The Importance of Assortment Changes in Historical Data

In order to calculate assortment elasticity, DT requires historical data that contains assortment changes, because DT examines historical data to determine how much cannibalization occurred when historical TAEs changed. From the relationship between changes in historical TAEs and changes in cannibalization, DT then calculates the assortment elasticity. This is similar to calculations of price elasticity. In order to determine price elasticity from historical data, it is necessary to have price changes in the historical data, and the more changes the better.

Using the example from "[Assortment Elasticity](#)", suppose, in the historical data for a particular store S, that the cookies assortment has one fewer SKU in week 10 compared to week 1 (that

is, some cookie SKU was removed). The TAEs for the other remaining cookie SKUs will all decrease between week 1 and week 10 because of the removal of the one SKU. DT then examines the changes in historical sales units of the SKUs in the cookie assortment at S between week 1 and week 10. By relating the changes in the sales units to the changes in TAEs, DT can calculate the assortment elasticity. A larger magnitude elasticity will result if the changes in TAE caused a large increase in sales units; a smaller magnitude elasticity will result if the increases are moderate.

In reality, the comparisons in this historical analysis that DT does are always more complex than in this simple example. It is rare to find a pair of weeks where the assortment change was just removal of a single SKU. Typically, in each pair of weeks, there are many assortment changes, involving both additions and removals, and the changes in TAE are a result of all of those changes. In the end, though, the relationship between the changes in TAE and the changes in sales units are summarized in a single number, the assortment elasticity, across all pairs of weeks. Because this single number summarizes the number of pairs of weeks and SKUs where TAEs changed, it is necessary an average over all the pairs of weeks and SKUs in the historical data and is not tuned to any particular SKU.

If Category Management Planning and Optimization (CMPO) is used to remove a single SKU from an assortment, it is likely that no pair of weeks in sales history exists in which exactly this SKU was removed and only this SKU was removed. For forecasting the results of this removal, CMPO makes an extrapolation from the historical analysis described above and uses the assortment elasticity that is not tuned to this particular situation of removing only this one particular SKU.

Estimation of Assortment Elasticity

One possible approach to the estimation of assortment elasticity is to estimate it jointly using base rates of sale for each SKU-store and seasonality parameters. However, the estimates of base rates of sale or of seasonality are not required, and an estimation technique that will estimate only assortment elasticity is used. Seasonality is taken into account by assuming that the entire category of items has similar seasonality at a given store, and then applying a differencing technique to get rid of seasonality, leaving base demand and assortment elasticity. For a particular SKU X at store S, the estimation uses pairs of weeks as follows: Suppose W1 and W2 are a pair of weeks where the assortment for the category at W2 at S is different from that at W1. The ratio of sales of X at W2 to sales of X at W1 is taken. This ratio is the dependent variable of the regression, and the independent variable is the assortment difference between W1 and W2. As explained above, the effect of the assortment on X is encoded as the TAE of X and this ratio of sales must be explained in terms of the ratio of the TAE of X between W1 and W2. The input to the regression is many such pairs of weeks for each SKU and each store, and the output of the regression is a single assortment elasticity over all the SKUs and stores of the input. The weeks in each pair are reasonably close together, so that the weeks are more likely to represent similar conditions and provide a better estimate. That is, with such weeks, the only change is likely the assortment change, and it is more likely that the difference in sales units is due to the assortment change than to other external factors. Estimation performs a weighted regression, with the observations from lower-selling SKUs weighted less than observations from higher-selling SKUs, because lower-selling SKUs have more volatile sales and thus should be weighted less according to statistical theory concerning linear regression. Note that since the estimation uses ratios of sales units of SKU X, the absolute quantity of the sales units does not matter (except for the weighting in the weighted regression) for the estimation. The removal of seasonality described above requires that the weeks W1 and W2 in a pair share common items in the assortment within a store. For the pair of weeks to be in the regression, the weeks must have at least five items in common, though this is configurable. Category-store combinations in which the store carries very few items from the category may not meet the requirement of five common items, and a value smaller than five may be necessary, depending on the importance of such category-store

combinations. Note that requiring fewer common items results in a poorer seasonality correction, and that five is already a very low number. For example, grocery clients for the demand-transference application usually have assortments numbering in the hundreds. The requirement for common items within-store means each store can have its own seasonality, though this requires a number of common items. In the above description of the regression, W1 and W2 were weeks, but in typical use, the regression uses longer time periods, with a default of four weeks. So W1 and W2 in the default configuration are four-week periods, and the regression uses the average sales of SKU X over those periods. This helps smooth out any noise in the sales of X, and also makes the regression more amenable to low-selling SKUs. Note that extreme low sellers, such as items that sell fewer than one unit in four weeks, will likely still present a problem for the regression.

The Meaning of the Possible Values of Assortment Elasticity

An assortment elasticity of 0 turns all cannibalization factors into constant 1, meaning the assortment has no cannibalization. This is highly unlikely. However, it does show that small-magnitude assortment elasticity indicates a category where cannibalization is small. Likewise, a high magnitude of assortment elasticity indicates a category where cannibalization is large. It is possible for the magnitude to be too large (see "[The Substitutable Demand Percentage](#)").

It is also possible for the Calculation Stage of DT to produce assortment elasticities that are positive. Such positive values for assortment elasticity are an indication that there is some unidentified problem with the data, because a positive assortment elasticity means cannibalization factors increase with increasing assortment size, which in turn means each item in the assortment sells more the larger the assortment becomes. This is presumed to be a nonsensical result, and, in the Evaluation Stage of DT, such positive assortment elasticities are removed and replaced by assortment elasticities from escalation (that is, the elasticities are replaced with higher-level ones).

It is possible, with sufficient data analysis, to figure out what problem with the historical data caused the positive assortment elasticity. However, such analysis is difficult to automate, and escalation is used instead.

The Substitutable Demand Percentage

The substitutable demand percentage, or just substitutable percentage, of an item in an assortment is the fraction of its demand that is retained by the assortment if the item is removed from the assortment. It is a measure of how substitutable the item is. For example, if the substitutable percentage is 100 percent, then removing the item will not decrease the total sales units of the assortment, since all of the demand for the item will transfer to the other items that remain in the assortment. If, on the other hand, it is 50 percent, then the removal of the item from the assortment means that 50 percent of its demand is lost, and 50 percent is retained. The total assortment sales units will decrease if this item were to be removed from the assortment.

The magnitude of the assortment elasticity has a influence on the substitutable percentage. Increasing the magnitude of the assortment elasticity increases the substitutable percentage. DT only calculates the assortment elasticity for the entire category (not per item), so changing the value of the assortment elasticity changes the substitutable percentage for all items in the category all at once.

It is possible for the magnitude of the assortment elasticity to be too large. This is indicated by a substitutable percentage for several of the items in the assortment that is over 100 percent. A few items can have substitutable percentages over 100 percent, because those are probably outliers. If the assortment is large, it is likely that a few such outliers exist. If 10 percent of items in the assortment are over 100 percent, then the results should be examined.

DT provides a tool for examining the substitutable percentage and for decreasing the assortment elasticity if too many items have a substitutable percentage over 100 percent. Here are some guidelines for using this tool.

- When selecting the time interval for the tool, select one that is likely to contain assortments that are representative of the retailer's current assortments. Since the retailer is going to be using the assortment elasticity in forecasts of what happens when current assortments are modified, it makes sense to test the assortment elasticity against assortments that are as similar as possible to the current ones.
- It is possible to use the tool to dial down the assortment elasticity. Using Setting maximum substitution percentage, DT calculates an assortment elasticity that results in substitution percentages that do not exceed the set maximum. When using this feature, you may want to set the maximum to a value higher than 100 percent if there are some outlier items that have high substitution percentages. Forcing these outliers down to 100 percent may result in a small-magnitude assortment elasticity, which may mean unacceptably small substitution percentages for all except the outlier items. You may want to select a maximum that is higher than 100 percent but that still brings most items down to 100 percent, leaving a few outliers above 100 percent.
- It is possible to use this tool to set the maximum percentage even if all substitution percentages are already below 100 percent. You may have business knowledge, or a directive from the retailer, and know that a particular category must exhibit a substitution percentage of at most 70 percent. In this case, this tool can be used to bring the substitution percentages down to 70 percent. This can make the difference between acceptance and rejection by the client.

No Requirement for a Time Interval

A time interval for the CDT calculation can be set in the CDT Data Setup stage. No equivalent exists in the Data Setup stage of DT.

The cannibalization factor directly incorporates information about the assortment through the TAE, and so the cannibalization model can handle large assortment changes. This makes it less necessary to use a time interval for DT, compared to CDT, because historical assortment changes can be directly accounted for in the model as changes in TAE.

Segments vs. Locations

In the Calculation Stage for both DT and CDT, it is possible to set up the calculation so that it is performed at all combinations of levels of the segment hierarchy and the location hierarchy. This is a more practical possibility for assortment elasticity than for the CDT calculation, because the assortment elasticity is not examined directly by people (unlike the CDTs), and producing thousands of values will not cause an issue. However, it is recommended to use only one of the two hierarchies in the Calculation Stage. Set either the segment hierarchy or the location hierarchy (or both) to be Chain. Because the calculation of assortment elasticity requires assortment changes in history, generating assortment elasticities at all levels may mean that, at lower levels, the data does not contain enough assortment changes in history. You may want to use your business knowledge of the particular retailer or particular category here, since you may know for the retailer or for the category whether assortment changes are frequent or not in the historical data you have. If the assortment changes are infrequent, you may be better off calculating a Segment- Chain/Location Chain assortment elasticity only.

Setting the Escalation Path

The last stage in DT involves setting the escalation path. If you are using only the segment hierarchy or only the location hierarchy, the escalation path is simply the hierarchy that you are using, and you set the escalation path according to the hierarchy. If you are using both a location hierarchy and a segment hierarchy, then usually you should set the escalation path to go up the segment hierarchy first, and then the location hierarchy. It is better to use only one of the hierarchies.

When using both hierarchies, the escalation path is necessary in order to tell the application which parent it should go to when moving up from a given segment/location node. With both hierarchies in play, every segment/location node has multiple higher-level nodes that do not lie along a single path. The escalation path is necessary to tell the application in what order the higher-level nodes should be considered. When only one hierarchy is used, the higher-level nodes form a single path.

Automatic Updating

DT can automatically and periodically update the assortment elasticities as new sales history is available. This feature is unique to DT; CDT does not perform automatic updating because it makes less sense to automatically produce new CDTs. New assortment elasticities can be loaded into the consuming applications and thus immediately used; however, the value of new CDTs is less clear.

When new historical transactions enter the RA schema, DT will automatically aggregate them and produce new SKU-store-week sales-units aggregates. These new aggregates are then appended to the older SKU-store-week aggregates, and the resulting data set is then used in a new calculation of assortment elasticities.

Note the following about this calculation:

- It does not in any way run the full DT application, that is, re-run all of the stages. The calculation is more targeted and just calculates assortment elasticity.
- It only updates assortment elasticities, not the similarities from the Similarity Calculation Stage.
- Because it uses a mix of old data and more recent data, the values of the assortment elasticities will change slowly over time as the data set becomes more tilted towards newer data. This is by design. It is not desirable to have sudden changes in assortment elasticity, since that would result in sudden changes in cannibalization and demand transference.
- Any assortment elasticities that were overridden using the Substitutable Percentage tool (see "[The Substitutable Demand Percentage](#)"), stay overridden, and are not updated.

Avoiding Categories with Small Assortments

It is possible for a retailer to have categories where the assortments are very small, that is, 20 or fewer items in the assortment. Such categories can pose a problem for DT because of the small amount of sales data for just 20 items, and also the number of assortment changes may be quite few.

It is better if the assortment is small but items from a much larger set have been added or removed frequently from the assortment. That is, the category has a much larger set of items, but only 20 of them are in an assortment at a given time. It is possible that the assortment

changes were frequent enough that more than 20 items have sales history, and in this case DT results may be reliable even though the assortment is small.

Implementing DT for Fashion Categories

For various reasons, fashion categories require some special consideration. This section describes what is different about them and how to handle the differences.

Proper Level for Fashion Categories

The lowest level of data must not be the SKU level, that is, the Size level of the merchandise hierarchy. Because size is a functional-fit attribute, or nearly so, the level of calculation must be at least one level above size (Style-Color). The historical sales-units data must be aggregated at least up to Style-Color. This also helps avoid problems with low sales rates and noisy sales rates at the SKU level, both common problems for fashion categories. It also helps decrease the number of SKUs within a category, since a multitude of sizes are possible for each Style-Color.

It is worth considering whether color is necessary. Aggregating to the Style level means that transference among colors cannot be calculated. However, it is not clear how useful calculating transference among colors would be, since the colors change for every selling season, and calculating historical transferences among colors may not be particularly useful. A possible halfway approach here might be to aggregate to Style-Primary-Color, where there are only a few primary colors. The primary colors chosen can be the ones that are stable season after season, so that historical transferences among them might be useful in future selling seasons. The primary colors can be chosen to be groupings of the actual colors (so for example, midnight blue and sky blue would become blue). In general, in fashion, the number of colors can be large, and it is unlikely that calculating transferences among such colors would be useful. Aggregating to Style-Primary-Color or even to Style can help avoid low sales rates and noisy sales rates.

It is possible to employ different approaches in grouping the colors. One is to use the primary color. It is also possible to group the colors based on the type of customer the color is designed to attract. For example, the colors can be grouped in "trendy colors" vs. "basic colors." The grouping should be decided in consultation with the retailer, to determine how the retailer uses colors in the category. The retailer may already have a grouping of colors that it uses, and the simplest approach may be to use this grouping.

Here, it is assumed that the aggregation of the historical sales-units data to be either Style-Primary-Color or Style. The term "item" should be understood to mean Style-Primary-Color or Style, depending on the chosen aggregation.

Typically, for fashion, the number of colors is large because of all the color variants. If it is necessary to retain all of the colors instead of following the recommendations above, then it will be necessary to split the color attribute into at least two attributes, a primary color and a secondary color. For more information, see "[The Role of Attributes in Calculating Similarities](#)".

Seasonality (Life Cycle) Considerations

DT makes the assumption that the items within a category at a store all have a common seasonality (see "[Seasonality in Historical Sales Data](#)"). Because of the short (tens of weeks) life cycles of fashion items, and because items within a category may have different introduction times within the same store, the assumption of common seasonality across the items in a category is probably not valid for fashion categories. It is possible within the category to have items that are at various points in their life cycles within the same store. At the same

time, some items may be in the uptrend part of their life cycle, while other items are in the downtrend part of their life cycle.

There are some ways to deal with this by properly setting up the input data for DT.

One simple approach is to approximate the life cycle of an item by using the SKU-store-ranging described in [Demand Transference](#). In this approximation, the range for an item is set to start at x percent sell-through of the item and end at y percent sell-through. Choose x to be 5, and y to be 70. X must not be 0 (that is, the point of introduction of the item), since it takes sometime after the point of introduction of the item for customers to start buying it in quantity and for the item to start having any kind of cannibalization effect on the other items. Y must be set to a point where significant numbers of customers have started to either lose interest in the item or where the item no longer has sufficient numbers of sizes available. In either case, customers are now transferring their demand to the other items in the assortment, so it is as if the item were no longer in the assortment. This is an approximation as the item is still in the assortment and is still selling, just at a significantly lower rate than its peak sales rate.

6

Advanced Clustering

This chapter describes the Advanced Clustering application. It provides details on configuration and implementation.

Overview

Advanced Clustering (AC) lets users create store clusters based on common features such as customer demographics in order to manage merchandise assortments and pricing strategies in a targeted way. Clusters can help retailers understand who shops in their stores and what their preferences are. Clusters can be used to inform decisions about assortment, pricing, promotion, forecasting, allocation, and supply chain processes based on selling patterns in stores. An understanding of the characteristics of the customers who shop in a store and what they buy can help a retailer target specific customers

The application optimizes clusters in order to determine the minimum number of clusters that best describes the historical data used in the analysis and that best meets the business objectives defined when the clusters are designed. Users can define a hierarchy cluster of stores based on a store attribute such as format and then cluster further using performance attributes in order to determine which stores have high, medium, and low sales. What-if scenarios and ranking can be used to compare how cohesive and well separated clusters are in each scenario as the number of cluster centers is increased. The application uses scoring to indicate which clusters fall below defined thresholds and may require manual intervention. Business Intelligence graphics illustrate the patterns in the data and the attributes that are important in each cluster.

The key features available in AC are:

- Dynamic nested clustering, in which a user can cluster on a criteria, analyze the results, and then decide whether or not to further sub-cluster.
- Mixed attribute clustering. A cluster can be created on continuous attributes such as performance (sales, revenue, and gross profit) as well as discrete attributes such as store size, demographics, and seasonality, all at the same time.
- Configurable clustering criteria such as customer profiles, product attributes, performance, and store attributes.
- Recommendations are made for the optimal number of clusters and the scores for each cluster. These are based on the quality of the clusters: how cohesive and well separated the clusters are.

Data Requirements

Advanced Clustering relies on following data and it uses ETL to load the data.

Table 6-1 Data Requirements

Objects	Granularity	Required/Optional
Hierarchies	Product, Location, and Fiscal	Required

Table 6-1 (Cont.) Data Requirements

Objects	Granularity	Required/Optional
Location Attribute	Store	Required
Product Attribute	SKU	Required
Aggregate Sales Data	Week/SKU/Store	Required
Customer Segment Profiles	Store/Customer Segment/ (Category or All Merchandise)	Optional
Alternate Hierarchy	CM Group or Trade Area	Optional
Like Locations	Store	Optional
Product Attribute Group and Value	Category or Subcategory	Optional
Aggregate Forecast Sales Data	Week/SKU/Store	Optional

Multiple Hierarchies and Level Support

Store clusters can be generated for the following combinations of hierarchies.

Product Hierarchies

This includes:

- Core merchandise hierarchy
- Alternate hierarchy

Clusters can be defined for either of the hierarchies and for different hierarchy levels. For example, clusters can be defined for Chain, Department, or Category along the product hierarchy.

Note that store clustering can only be defined for a product hierarchy level higher than Item. Item level store clustering is not supported. However, store clusters can be generated for item groups by defining an item group as a level in the alternate hierarchy.

Location Hierarchies

This includes:

- Core location hierarchy
- Alternate hierarchy (optional)

Clusters can be defined for either of the hierarchies and for different hierarchy levels. For example, clusters can be defined for Chain, Trade Area, or Region along the location hierarchy. Store clusters can be generated for channel, if channels are configured as a level in the location hierarchy.

Calendar Hierarchies

This includes:

- Core fiscal calendar hierarchy (week, month/period, quarter, half, year)
- Gregorian calendar (week, month, quarter, half, year). Leverages a start and stop date (day level date range)
- Planning period. Leverages alternate hierarchies, including planning period, buy periods, and defined holiday time periods such as back to school and fourth of July. This is optional.

Clusters can be defined for any of these three calendar hierarchies (the cluster effective period). Note that the source time period for historical data only uses the core fiscal calendar hierarchy, supporting hierarchy levels that include week, month/period, quarter, half, and year.

Clustering Criteria Supported in Store Clustering

In store clustering, the cluster criteria are a set of attributes that define store clusters. These attributes can be either discrete or continuous. A group of these clusters is called "Cluster by." For example, demographic data such as income and store properties such as store format can be grouped into a store attribute Cluster by.

These default Cluster by are supported.

Customer Profile

Stores are clustered based on the similarity in the mix of customer profiles shopping in the stores and trading areas. These clusters form the basis for further analysis to understand which customers shop in which stores and how they shop. Retailers obtain market data from market research firms such as the Nielsen Corporation and use the data to create customer profiles for their stores. An application feed can be used to provide this information to AC at the category or all merchandise level.

Location Attribute

Stores are clustered based on how shopping behavior varies by store attribute. This provides information about who is shopping in a store or trading area as well as demographic data such as ethnicity, income levels, education, household size, and family status. Retailers can analyze the cluster composition and related business intelligence in order to better understand the shopping behavior of their customers. This can help retailers make assortment and pricing decisions.

Product Attributes

In this type of clustering, stores are grouped together that have similar sales shares for one or more product attributes (for example, coffee brands such as premium, standard, and niche). The percentage of each store's contribution is calculated using the Sales Retail \$ for each product attribute value to the total retail sales of the category or subcategory of each location.

Product attributes can be configured only at category or sub-category levels.

Performance Criteria

Stores are clustered based on historic sales by considering performance at different merchandise levels while performing store clustering and analyzing how the shopping behavior varies by category. This can help to identify high, medium, and low volume stores.

Forecast Criteria

Stores are clustered based on forecast sales by considering future sales at different merchandise levels while performing store clustering and analyzing how the shopping behavior varies by category. This can help to identify high, medium, and low volume stores based on the predicted sales.

Mixed Criteria

Discrete and continuous attributes are combined together. Retailers can cluster stores using attributes from all the above defined criteria at the same time.

Price Elasticity

Stores are clustered based on their price elasticity, by considering if it is for markdowns or promotions and obtaining the latest approved PMO run mapped to AC run type that runs Short Life Cycle. Note that this gets the root nodes for the flex group, customer segmentation, and price zone. This has no time dimension considered.

Attributes in Store Clustering

Cluster by uses a collection of attributes, including consumer profile attributes, sales metric attributes, location attributes, and product attributes.

Sales Metrics

Store clustering uses a fixed set of sales metrics. These attributes cannot be extended. Supported attributes include Sales Retail \$, Sales Unit, Sales AUR, Gross Margin R, and Gross Margin %.

Forecast Sales Metrics

Store clustering uses a fixed set of predicted sales metrics. These attributes cannot be extended. Supported attributes include Forecast Sales Retail \$, Forecast Sales Unit, Forecast Sales AUR, Forecast Gross Margin R, and Forecast Gross Margin %.

Location Attributes

Store clustering relies on location attributes that can loaded into the application as either core attributes or as user-defined attributes. These attributes are defined for stores. They define the store properties, including demographic and geographic details. During installation, only attributes that have 15 distinct values are configured. Attributes with higher discrete values are not considered for store clustering.

Product Attributes

Product attributes based on store clustering use two types of attributes, raw attributes and grouped attributes. The former are product attributes, identified as important attribute values. The latter are fed into the application and are available to the store clustering process only when the CDT application is enabled. The clustering process groups together stores that have similar attributes values for a product category. A store share is calculated using Sales Retail \$, which is the ratio of the sales retail of each product attribute value to the total sales retail of the category or subcategory of the specific location.

Each category or subcategory must have raw attributes and grouped attributes.

Grouped attributes (the default) classify the items in the class or subclass. This set of attributes differs from class to class. For example, for yogurt, the attributes are: size, flavor, brand, fat percentage, and pack size. For chocolate, the attributes are: size, brand, milk/dark, nut type, and package type. The two classes can both have the attribute of brand, but the brand attribute will have different values for each of the categories. Group attributes have a mapping of each item in the category to its set of attribute values. This information is provided as a data feed to the application. If grouped attributes are not available then raw attributes are used for the store clustering of product attributes.

Raw attributes typically have a large number of attribute values. For example, the brand attribute for yogurt may list 50 different brands at a large grocer. Using raw attributes, the system runs a preprocess to identify n (default = 3) attributes values that are most frequently sold for each attribute in a category or subcategory.

Configuration Process

Default configuration occurs during the installation and upgrade. The configuration process is responsible for installing or upgrading any new attributes in the application. This process ensures that any existing manual overrides introduced by the retailers are not overridden and any new additions are brought into the clustering process. The default configuration includes the following:

- All attributes are enabled by default and weights are normalized among all the configured attributes.
- Any discrete location attribute that has more than $n=15$ attributes values is not configured by default. Note that the value of n is a configuration and can be modified at the time of deployment.
- The UI formatting of each attribute is identified based on the data type of the attributes.
- Nesting is enabled by default for all types of Cluster by (except mixed, which is an alternative approach to clustering).
- The deployment of clusters at multiple hierarchies or levels is enabled.

The following configurations may require manual overrides if the default configuration is not acceptable or data is not available.

Table 6-2 Manual Overrides

Name	Description
Enable or disable Cluster by	Disable Cluster by. For example, for the consumer profiles for each store for a retailer, the Cluster by for the consumer segment must be turned off.
Enable or disable nesting	Allow multiple nesting levels under an existing Cluster by. For example, cluster first by product performance, with nested clustering by store attributes.
Enable or disable attributes	Enable an attribute to be considered for clustering or contextual BI. For example, label the population density attribute as a BI attribute instead of a clustering attribute, as very few stores have data for population density, and it is not significant enough for store clustering.
Change UI formatting	Change formatting associated with the attributes, such as label, decimals, percent, and currency. These are configurations for each attribute and do not rely on XLIF entries.
Cluster deployment	Enable or disable hierarchy at which clusters can be deployed. For example, if CMPO store clusters are only defined for categories, then only clusters at the category level will be approved. The other levels can be enabled if needed.
Outlier Rule	Change default outlier rules for a Cluster by. By default, the distance from centroid rule is enabled. See section below for other supported outlier rules.
New Store Rule	Change default store rules for a Cluster by. By default, the like location rule is enabled. See section below for other supported new store rules.

Table 6-3 Enable or Disable Cluster By

Cluster By	Description	Example	Enable
Customer segment profile	Cluster store using consumer segment distribution	20% soccer mom, 30% empty nesters	Enable if consumer segment profiles are available for each store.
Store attribute	Cluster store using location attributes	Income, climate, size, store format	Enable if location attributes are available for each store
Performance	Cluster store using sales metrics	Sales revenue, sales unit, gross margin \$	Enable if retailer wants to cluster store using performance metrics
Product attribute	Cluster store using product attribute sales shares	Brand, color, seasonality, size/fit	Enable if retailer wants to cluster store using product attributes sales share
Mixed attributes	Cluster store using mixed attributes, combining attributes across all the cluster criteria	Income, climate, size, store format, sales revenue, sales unit, gross margin	Enable if retailer wants to cluster store using combination of attributes
Product forecast	Cluster store using predicted sales metric	Forecast sales revenue \$, forecast sales unit, forecast gross margin \$	Enable if retailer wants to cluster store using future sales metrics

Copy Clusters Using Like Product Mapping

Advanced Store Clustering supports copying clusters from other products by defining like product mapping. Some of a retailer's products, such as tea, may not be mature enough, so the retailer must copy clusters from a mature category such as coffee. The solution copies clusters that have already been approved from that category to a new category and then make the clusters available to the execution solution during export. This feature also supports the requirement in which categories in certain departments may want to borrow clusters from a higher product hierarchy such as Department or Division. In both use cases, clusters must be borrowed from another product (department or category) with minimal effort by the end user to define and approve the cluster criteria.

The retailer can define a like product interface that the retailer can use to provide mapping between a mature product and a new or poorly performing product. This product mapping can vary by location (for example, the tea category mapped to coffee in Trade Area East, and mapped to other beverages in Trade Area West).

This category mapping is selected automatically during export, based on the like products, and is assigned clusters from the mapped category that have already been approved by the end user. Cluster copy will respect product mapping if it is updated or removed using the effective period, and exports will be generated accordingly. Retailers can review the cluster composition of the mapped category via the UI from which it is borrowing the clusters; however, the copied clusters for the category will not be available for the user to review in the UI.

The Copy Clusters feature using like product mapping is only supported for the basic Store Clustering interface that is available for Oracle Retail Predictive Application Server applications such as Category Management; it is not supported as part of the Store Clustering generic interface or the REST API.

Multiple Clustering Approach

Store clustering supports three types of clustering: simple, nested, and mixed. By default, all three approaches are enabled. These approaches are applicable to all Cluster by. Store clustering functionality supports dynamic nesting capabilities. For example, the user can cluster on a criteria, analyze the results, and then decide to further sub-cluster.

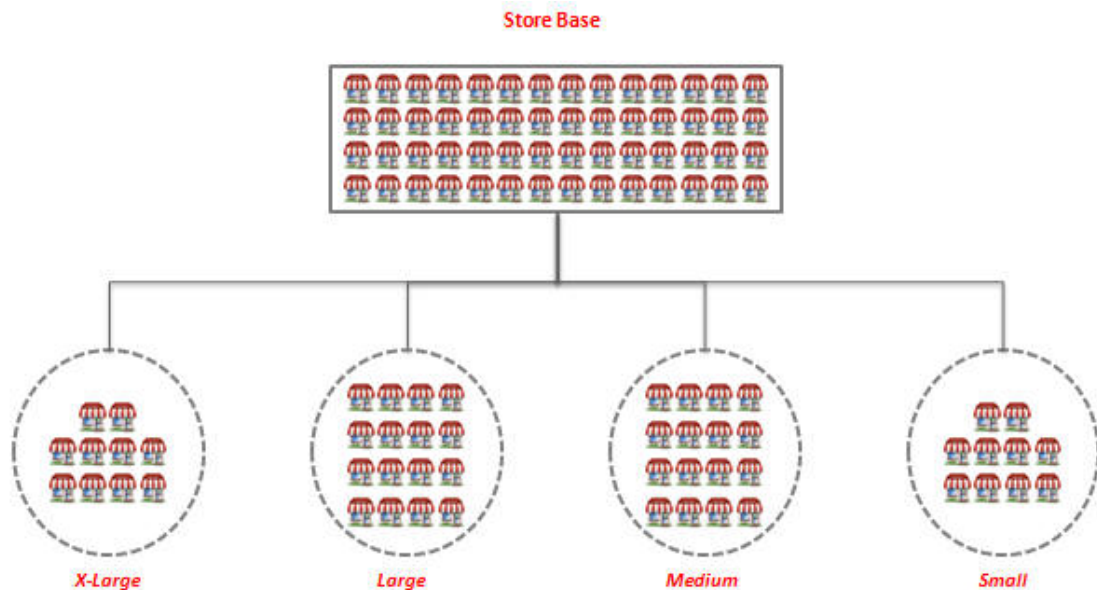
Mixed attribute clustering is also supported. For example, it is possible to cluster on continuous attributes such as performance (sales, revenue, and gross profit) as well as discrete attributes (store size and demographics) at the same time.

Simple

Users can select attributes from a Cluster by. For example, users can select location attribute Cluster by and generate clusters using location attributes such as store size.

Users select the most important store attribute (based on category) and group the stores accordingly. Clusters may or may not cross trade areas, regions, and districts. This depends on the approach as well as the responsibility of the planning team.

Figure 6-1 Simple Clustering



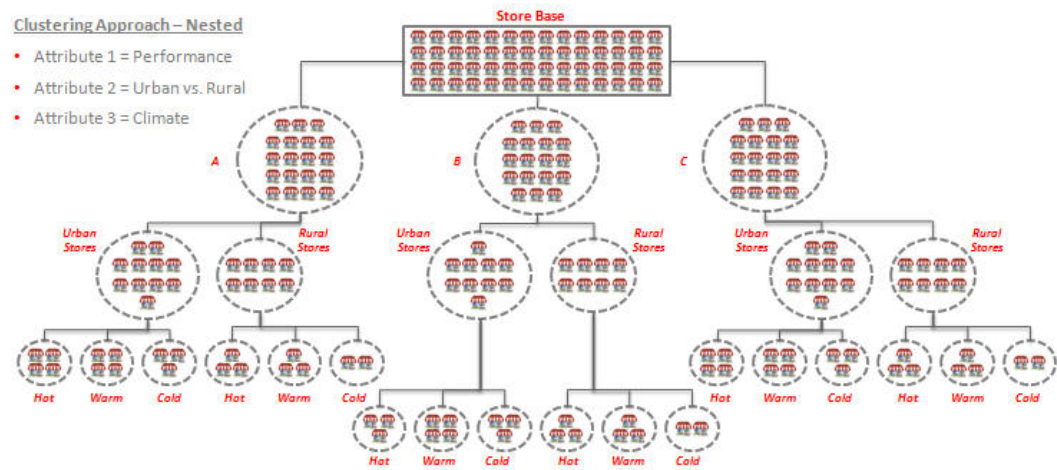
Nested

All Cluster by, except Mixed Attributes, can use a nested hierarchy by default. For example, performance clusters can be further clustered using location attributes. This approach allows a dynamic hierarchy of clusters. Nesting can be enabled or disabled in the AC configurations.

Users can select the most important store attributes (based on the category and the group of stores selected during the wizard process) and group the stores accordingly to ensure that a more refined assortment can be created by category or location selection. Clusters may or may not cross trade areas, regions, and districts. This depends on the approach as well as the responsibilities of the planning team. Once the initial clusters are created, users can further cluster using attributes to define nested clusters. This can be done within a trade area as well

as at the total company level. The number of clusters is granular, as compared to mixed attributes.

Figure 6-2 Nested Attribute Clustering



Mixed

Mixed attributes, including Cluster By such as consumer segment, location attributes, performance, and product attributes, are supported by default. Users can combine attributes from different Cluster by. For example, users can combine attributes from consumer segment and performance Cluster by and generate clusters using sales revenue and consumer segment distributions.

Users can use the most important store attributes (based on the category and the group of stores selected during the wizard process) and group the stores accordingly. Clusters may or may not cross trade areas, regions, and districts. This depends on the approach as well as the responsibilities of the planning team. These clusters are the final ones used in assortment process. The number of clusters in this case is confined, compared to the nested clustering approach.

Figure 6-3 Mixed Attribute Clustering



New Stores or Stores with Poor History

Store clustering supports three rules for allocating new stores to a cluster. These rules can be configured for each Cluster by. A rule is applied after the clusters are generated.

These rules require a feed into the application that defines a mapping between a location and like locations. This mapping can be configured by merchandise. One location can be mapped to multiple locations with different weights.

Like Stores (Default Rule)

Stores with new history or poor history are allocated to the same cluster in which the like locations are allocated. For example, a new store or a store whose poor history has been corrected can be allocated to a valid performance cluster.

Largest Clusters

New stores or stores with a poor history can be allocated to the largest cluster identified by the clustering analytics. For example, a new store that has not yet formed a customer base can be allocated to a larger cluster until significant customer profiles have been collected.

Cohesive Clusters

New stores or stores with poor history are allocated to the most compact cluster identified by the clustering analytics. For example, stores can be assigned to a cluster that has not been not affected by outliers.

Outliers

Store clustering supports two rules that identify stores as outliers in a cluster. These rules can be configured for each Cluster by. The rule is applied after the clusters are generated.

Distance from Centroid (Default Rule)

The distance from a store to the centroid is identified. If the distance is beyond a defined limit of the configured threshold from the centroid, then the cluster is identified as having outliers. The user must investigate such clusters.

Cluster Size

The percentage of stores that are allocated to certain clusters is identified. If they fall beyond a certain limit in comparison to total number of stores, the cluster is identified as having outliers. The user must investigate such clusters.

Export to Excel

To use any reporting tool with an Excel file exported from Advanced Clustering, you may need to adjust the format. Here are some examples of possible formatting adjustments.

- The Text column should remain as is.
- If the Percentage column uses a percent symbol, then that symbol must be removed.
- If the Currency column uses either commas or currency symbols, then those must be removed.
- If the Number column uses commas, then they must be removed.

Configuring Cluster Exports

Cluster export jobs are used to send cluster information to other applications, such as Assortment Planning, and so on. By default, cluster export jobs are scheduled to run weekly. The frequency of the exports can be configured to run daily, if required. This can be done for a small portion of the Advanced Clustering (AC) exports, or for all the Advanced Clustering exports.

The following steps must be executed to apply this configuration.

To configure all the AC Exports for daily export:

1. Log into the application.
2. Navigate to Control & Tactical Center > Strategy & Policy Management > Manage System Configurations.
3. In the Table drop-down menu, select **RSE_EXP_GRP**.
4. Search for the record with these details:
 - a. EXP_GRP = AC
 - b. NAME = Advanced Clustering Exports
5. Select the record and click **Edit**.
6. Update the PARENT_EXP_GRP value from WEEKLY_BATCH to DAILY_BATCH.
7. Click **OK**.

To configure only a subset of the AC Exports for daily export:

1. Log into the application.
2. Navigate to Control & Tactical Center > Strategy & Policy Management > Manage System Configurations.
3. In the Table drop-down menu, select **RSE_EXP_GRP**.
4. Search for the record with these details:
 - a. EXP_GRP = AC
 - b. NAME = Advanced Clustering Exports
5. Click **Create** or the **+** icon.
 - a. Enter the necessary details. (You can change the following values.)
 - i. ID : [Max RSE_EXP_GRP.ID + 1]
 - ii. EXP_GRP : AC_DLY
 - iii. NAME : Advanced Clustering Export Daily
 - iv. RELATIVE_DIRECTORY :
 - v. PARENT_EXP_GRP : DAILY_BATCH
 - vi. FROM_DT :
 - vii. TO_DT :
 - viii. EXP_TYPE :
 - ix. LEAF_NODE : N
 - x. EXP_VIEW_NAME :

- xi. EXP_TABLE_NAME :
- b. Click **OK**.
- c. Choose the AC exports to be reconfigured to Daily.
- d. Change the PARENT_EXP_GRP of each to the newly created EXP_GRP (i.e. AC_DAILY).
- e. Look for the AC export to be reconfigured.
- f. Click **Edit** or the pencil icon.
- g. Change PARENT_EXP_GRP from WEEKLY_BATCH to AC_DLY.
- h. Click **OK**.
- i. Repeat the process for any other AC exports to be reconfigured.

7

Customer Segmentation

This chapter provides details about the use of the Customer Segmentation application.

Overview

Customer Segmentation (CS) lets users create segments of customers based on common attributes, such as customer demographics, in order to help a retailer manage merchandise and sales strategies in a targeted way. Segments can help retailers understand the types of customers who shop in their stores and gain insight into their typical shopping patterns. This understanding can help retailers target specific customers.

The application optimizes segments in order to determine the minimum number of segments that best describes the data used in the analysis and that best meets the business objectives defined when the segments are designed. What-if scenarios and ranking can be used to compare how cohesive and well separated the segments are in each scenario as the number of segments is increased. The application uses scoring to indicate which segments fall below defined thresholds and may require manual intervention. Business Intelligence graphics illustrate the patterns in the data and the attributes that are important in each segment.

The key features available in CS are:

- Recommendations are provided for the important attributes to use in creating segments.
- Segments can be created on continuous attributes such as sales performance as well as discrete attributes such as customer gender.
- Configurable segmenting criteria such as demographics and RFM (Recency and Frequency Measures) are provided.
- Recommendations are made for the optimal number of segments and the scores for each segment. These are based on the quality of the segments: how cohesive and well separated the segments are.

Data Requirements

Customer Segmentation relies on following data, and it uses ETL to load the data.

Table 7-1 Data Requirements

Object	Granularity	Required/Optional
Hierarchy	Product, Location, and Fiscal	Required
Customer Attribute	Demographic or User Defined	Required for demographic segmentation
Sales Transaction	Customer-identified, Product, Date, Transaction ID	Required for RFM segmentation
Alternate Hierarchy	CM Group or Trade Area	Optional

Multiple Hierarchies and Support

Customer segments can be generated for the following combinations of hierarchies.

Product Hierarchy

One of the following hierarchies can be used:

- Core merchandise hierarchy
- Alternate hierarchy

The level at which segments should be created can be configured. The user interface is used to create the specific node identified. This allows the creation of different segments of customers for different geographic regions. For example, different segments can be defined for Canada vs. France.

Location Hierarchy

One of the following hierarchies can be used:

- Core location hierarchy
- Alternate hierarchy (optional)

The level at which segments should be created is configurable. The user interface is then used to create the specific node identified. This allows the creation of different segments of customers for different geographic regions. For example, different segments can be defined for Canada vs. France.

Calendar Hierarchy

This includes:

- Core fiscal calendar hierarchy (week, month/period, quarter, half, year)
- Gregorian calendar (week, month, quarter, half, year). Leverages a start and stop date (day level date range)
- Planning period. Leverages alternate hierarchies, including planning periods, buy periods, and defined holiday time periods such as back to school and Fourth of July. This is optional.

Segments can be defined for any of these three calendar hierarchies (the segment effective period). Note that the source time period for historical data only uses the core fiscal calendar hierarchy. A configuration permits data aggregation at either the fiscal period or fiscal quarter, so the user can select any level that is at that level or above in the user interface.

Supported Segment Criteria

In customer segmentation, the segment criteria consist of a set of attributes that define customer segments. These attributes can be either discrete or continuous. A group of these segments is called "Segment by." For example, demographic data, such as income and gender, and store properties, such as store and format, can be grouped into a Customer Demographic Segment by.

Here is the list of the default Segment bys that are supported.

Customer Demographics

Customers are segmented based on the similarity in the values of the various customer attributes. Examples include gender, income, educational background, age, and range. Additionally, it is possible to use several user-defined numeric or discrete attributes.

RFM and Customer Behavior

Customers are segmented based on attributes that are comprised of aggregate metrics regarding their purchase behavior. Examples include the number of purchases, amount of sales, average basket size, and share of products purchased while being promoted. Retailers can analyze the segment composition and related business intelligence in order to better understand the customer shopping behavior associated with the segments.

Category Purchase Behavior

Customers are segmented in a similar manner to RFM and Customer Behavior, with the difference that these attributes are calculated for a selection of the most important product categories. The category level used here is configurable. It helps enable segmentation by product category purchase behavior.

Customer Segmentation Attributes

Segment by uses a collection of attributes, including demographics, purchase behavior, product purchase behavior, product profiles, and user defined. Each quarter, the batch processing creates new versions for each location at the configured level of the location hierarchy. During this process, attributes are summarized and their data is analyzed for usefulness for segmentation. An attribute can have a different level of usefulness for each of the different versions. For example, if the majority of customers in the Canada location provide a gender attribute, and the majority of the customers in the France location do not provide a gender attribute, then gender can be used in Canada, but not in France.

Furthermore, within the same location, it is possible for an attribute to be considered useful for the most recent quarter, while in the prior quarter it was not useful because there were insufficient values available during that time. The aggregate statistics about the attributes for a version can be seen using the Explore Data screen after selecting the segment criteria. Segments created in the previous quarter have different statistics than those that are created during a different quarter.

Demographics

Demographic segmentation relies on customer attributes that are loaded into the AIF data warehouse. The set of attributes used from the data warehouse customer dimension is fixed. If alternative attributes are needed, see [User Defined](#). Once loaded using AIF DATA jobs in POM, they can be used by Customer Segmentation. They define details about each customer that can be used for creating customer segments using those values. During the time when a new version is created, only attributes that have 15 or fewer discrete values are used. Attributes with a higher number of discrete values are not considered for customer segmentation.

Purchase Behavior

Customer segmentation uses a fixed set of sales transaction metrics, which are obtained from sales transactions identified by a customer ID. The values include Sales Quantity, Sales Retail \$, Gross Margin \$, Promotional Sales Quantity, Promotional Sales Retail \$, Promotional

Gross Margin \$, Number of Transactions, Average Transaction Count per configured period, SKU count, Transaction Basket Size, and Promoted Sales Share of Total Sales.

Product Purchase

Customer segmentation uses a selection of the top categories to portray the shopping patterns for each customer for some product categories. Each time a version is created, the categories with the highest amount of sales are picked as the categories for which Product Purchase based assessments are done. These attributes are similar in concept to the Purchase Behavior segment; however, these are specific to the top categories for the location associated with the version. The attributes include Sales Quantity, Sales Retail \$, Gross Margin \$, Promotional Sales Quantity, Promotional Sales Retail \$, Promotional Gross Margin \$, Number of Transactions, Average Number of Transactions, SKU Count, Transaction Basket Size, Promoted Sales Share of Total Sales, Average Basket Sales \$, and Average Basket Gross Margin \$.

Product Profile

Customer Segmentation uses a fixed set of category-based sales profile values. For the same set of top categories described in [Product Purchase](#), the share of the customer's total purchases for the category is calculated. The share of the Promotional Sales Retail \$ of the customers total sales is also calculated.

User Defined

For any attribute that is available for the customer, but is not accounted for in the default set of attributes, there are provisions for loading a set of customer or user defined attributes into the AIF data warehouse. These attributes can be either numeric values or discrete values. If the attribute value is numeric (such as a zip code), but must be treated as discrete rather than a ranged numeric value, then the attribute must be loaded to an appropriate text attribute column in RI. Any attribute that has more than 15 distinct values will not be used by the segmentation process.

Once an attribute is defined, it is possible to adjust the configuration data in the database to assign a more context-suitable name for the attribute. This enables the user interface to identify the attribute as a specific attribute, and not just as a generic Custom Text Attr or Custom Number Attr.

Configuration Process

Default configuration occurs during the installation and upgrade. The configuration process is responsible for enabling or disabling any attributes in the application. This ensures that the desired attributes are available for use during the segmentation process.

- All attributes are enabled by default.
- Any discrete attribute that has more than $n=15$ attributes values is not configured by default. Note that the value of n is a configuration and can be modified at the time of deployment.
- The UI formatting of each attribute is identified based on the data type of the attributes and by the name of the attribute.

The following configurations may require manual overrides if the default configuration is not acceptable or data is not available.

Table 7-2 Manual Overrides

Name	Description
Enable or disable Segment by	Disable Segment by. For example, if there are no customer demographic attributes, then the Customer Demographics segment by can be disabled.
Enable or disable attributes	Enable an attribute to be considered for segmentation. For example, if there is no intention on loading user defined attributes, they can all be disabled so that when new versions are created, no processing time is spent analyzing those attributes.
Change UI formatting	Change formatting associated with the attributes such as label, decimals, percent, and currency. These are configurations for each attribute, and do not rely on XLIF entries.
Location hierarchy level	If customer segments are desired for each location at a given location level, then the configuration can be adjusted so that all processing is done at that level. This also requires an adjustment of the approval level so that it allows the segments to be approved.
Outlier rule	Change default outlier rules for a Segment by. By default, the distance from centroid rule is enabled. See the section below for other supported outlier rules.

Table 7-3 Enable or Disable Segment By

Segment By	Description	Example	Enable
Customer Demographics	Segment customers using demographic values	Age range, income, gender	Enable if consumer demographic attributes are available
RFM and Customer Behavior	Segment stores using location attributes	Income, climate, size, store format	Enable if location attributes are available for each store
Category Purchase Behavior	Segment stores using sales metrics	Sales revenue, sales unit, gross margin	Enable if retailer wants to segment stores using performance metrics

Attribute Preprocessing

Before customer segments are created, the available customer data must be preprocessed in order to identify the sample of customers to use for segmentation and to determine which attributes are the most beneficial for use while creating segments. A few configurations can be manipulated to help improve this process. The following table defines some configurations that can be adjusted to control how attributes are used by the system. These values can be manipulated in the table CIS_TCRITERIA_ATTR.

Table 7-4 Attribute Configuration

Column	Description
DELETE_FLG	Set value to Y to prevent an attribute from being used by any processing.

Table 7-4 (Cont.) Attribute Configuration

Column	Description
SAMPLE_FLG	Set value to Y so that the attribute to be used has a stratified sample of customers. This should help ensure that an appropriate selection of customers are represented in the sample. Up to three attributes can be set to Y. If no attributes are configured with a Y value, then a random sample of customers will be used.
DISPLAY_FORMAT_ID	This can be adjusted to use different display formats, as defined in RSE_DISPLAY_FORMAT.
NUM_BUCKETS	Allows a different number of buckets for use when showing summary metrics for a numeric attribute.
ATTR_IMP_FLG	When set to a Y value, the importance of this attribute is analyzed for usefulness during segmentation. If an attribute will never be used for segmentation, setting the value to N will exclude it from the attribute importance calculations.

Another configuration that can be adjusted is the CIS_BUS_OBJ_TCRITERIA_ATT_XREF.VALIDATING_ATTR_FLG. For any attribute that has a Y value for this column, the attribute is shown in the Insights portion of the UI. If it is determined that an attribute should not be displayed, then this value can be changed to a value N so that it is excluded from display in the Insights results.

Segmenting Approach

Customer segmentation uses Oracle Data Mining for the creation of its segments. The k-means approach that is used results in the creation of segments in a hierarchical manner. The process automatically determines which attribute is the best attribute to split into an additional segment. This process is continued until the desired number of clusters has been achieved.

Customer Segment Store Profile Generation

Customer Segmentation calculates the sales share of customer segments for each store. These store profiles can be generated by the user for the approved customer segments from the user interface. They can then be consumed by RI to generate business reports and Store Clustering to generate customer centric store clusters.

Preprocessing

As part of preprocessing when a version is created, Customer Segmentation first filters the customer and then samples the customer.

Filter Customer

The filtering step allows the implementer to set the following conditions in order to remove outlier customers during different phases of Customer Segmentation,

Table 7-5 Filter Step Conditions

Filter Rule	Operation
Fake Customers	This rule discards customers if the number of transaction per day exceeds the value <i>x</i> (the max daily transaction threshold to identify a fake customer - default 10)
Discarding Customers	Keep customers where customer sales fall between min (sales or # of transaction) and max (sales or # of transaction). Minimum/Maximum percentile for amount of sales transactions. (default .001) Minimum/Maximum percentile for number of sales transactions. (default .001)
New Customer Rules	Include new customers if there is enough existing sales history for the customer. Use configurations that control the min/max values for average transaction count and average transaction amounts used to include new customers: Minimum/Maximum percentile for amount of sales transactions. (default .001) Minimum/Maximum percentile for number of sales transactions. (default .001)
Top Categories	A configuration that can limit the number of categories (<i>n</i>) that a user picks. The system picks it based on the default - Sales Revenue of the category to be used to calculate the top categories. Allowable values are limited to SLS_AMT, SLS_QTY, PROFIT_AMT, and SHARE variations of the same (i.e., SLS_AMT_SHARE).

Sample Customer

The sampling step allows the implementer to enable and adjust sampling customers.

Table 7-6 Sample Step Conditions

Sample Rule	Operation
Target Sample	Set the value to Y so that the attribute to be used has a stratified sample of customers. This should help ensure that an appropriate selection of customers is represented in the sample. Up to three attributes can be set to Y. If no attributes are configured with a Y value, then a random sample of customers will be used. This can be adjusted by changing CIS_TYPE_CRITERIA in Data Management UI.
Sample Size	The sampling percentage for Customer Segmentation. This can be adjusted by changing RSE_CONFIG in the Data Management UI.

Customer Metrics

The Retail Science process supports the following derived customer metrics, which can be used for analysis by an application such as Customer Engagement. These metrics are calculated using customer-linked transaction data. In addition to helping understand how customers have behaved in past, these metrics can also help predict future behavior.

Table 7-7 Customer Metrics

Metric	Description
Customer Latency/Time Between Visits	The number of days between each of a customer's transactions sales or return.
Customer Lifespan	The time between a customer's first and last purchase.

Table 7-7 (Cont.) Customer Metrics

Metric	Description
RFMThe	RFM (recency, frequency, monetary) score determines quantitatively which customers are the best ones by examining how recently a customer has purchased (recency), how often the customer purchases (frequency), and how much the customer spends (monetary).
Projected next purchase date	Prediction of the next likely customer purchase date.
Location Loyalty	How loyal are customers to a specific location? A value of 100% indicates that they always shop at a particular location.
Style Loyalty	How loyal are customers to a particular style? A value of 100% indicates that they always prefer one specific style.
Color Loyalty	How loyal are customers to a particular color? A value of 100% indicates that they always prefer one specific color.
Brand Loyalty	How loyal are customers to a particular brand? A value of 100% indicates that they always prefer one specific brand.
Price Efficiency Loyalty	How efficient are customers in getting a promotion price? A value of 100% indicates that the customer always buys items on promotions or is very efficient in obtaining a good price.

8

Affinity Analysis

This chapter describes the Affinity Analysis application.

Overview

Affinity Analysis (AA) is used to gain insights into customer shopping patterns. A key component of AA is the process of Association Rule Mining (ARM). This process examines sales transaction data and identifies associations between types of products. Such information can help a retailer understand that promoting one product is sufficient to help drive sales of another product, given the sales associations they exhibit. The processing of these algorithms occurs each week as part of the weekly batch execution, and a set of output files are provided to expose the association rules that have been discovered by the process.

Data Requirements

AA relies on the following data elements. These must be provided via text files, which are then loaded.

Table 8-1 Data Elements

Object	Notes	Required/Optional
Product Hierarchy	The ARM processing mainly operates at Sub Class, but it can be configured to different levels.	Required
Location Hierarchy	-	Required
Fiscal Calendar	-	Required
Sales Transactions	Must contain transaction IDs as part of the data. If the transactions include Customer ID, then customer segment results are possible.	Required
Customer Segments	Customer IDs and their association to a segment allows customer segment-specific results.	Optional
MBA_ARM_SRVC_LOC_STG	Can be used to limit the scope of locations processed, or to specify a set of locations to exclude from processing.	Optional

In order to calculate association rules, it is necessary to receive sales transaction data that include a transaction ID. This is used to identify which products were purchased by a customer as part of a single transaction. If the customer transactions also include a customer ID to identify the customer who purchased the transaction, and a customer segment dimension is provided that links customer IDs to customer segments, then it is possible to provide some results for each customer segment.

MBA_ARM_SRVC_LOC_STG

When specifying which locations to process or which locations to not process, the MBA_ARM_SRVC_LOC_STG interface can be used to limit the scope of locations to be processed. The data in this interface can be at any level of the location hierarchy. A customer may want to limit the scope of locations for the following reasons.

- Improve performance by only sampling some locations.
- Exclude locations that contain many wholesale transactions, where the transactions contain data for more than a single customer.
- Exclude locations that are experiencing a significant interruption to their normal sales pattern (for example, when undergoing a large scale renovation).
- Exclude locations that normally do not include customer-linked transactions from the ARM_PH_CS implementation, since suitable data to include for processing will not be available.

The SRVC_NAME column of this interface allows the specification of the service that must be filtered. If, however, all executions must have the same set of locations, then this column can be provided as a NULL value. The effect will be to use the same dataset for all the services. If, however, it is necessary to have some services use a different set of locations, then it is possible to provide the data specific to the different services. If data is provided for a SRVC_NAME, then the data must be provided with a SRVC_NAME specified. The valid SRVC_NAME values that can be provided are: ARM_PH (Product Hierarchy results), ARM_PH_PROMO (Product Hierarchy with Promotions results), and ARM_PH_CS (Product Hierarchy and Customer Segment, with Promotions results).

MBA_ARM_SRVC_CONFIG_STG

It is possible to provide a configuration file that will be used as the configuration for all automated batch runs, as well as the default values for any user-created runs. The expected process flow for this interface is that a user initially uses the UI to create user runs for the various types of processing that is supported. These runs must be for the same configuration levels that the application is configured to use. The user creates runs for some sample weeks, and gradually adjusts the various configuration points, until a run is created with an acceptable amount of output.

Once this has been done for each of the different types of calculations, then the configuration values that were used can then be provided via this interface so that all automated runs will use these settings.

If the system is configured to run the calculation at a level of the location hierarchy other than the entire company, such as chains instead, and it is necessary to have different configuration values for each of those chains, then a different set of values can be provided here for each location node.

The SRVC_NAMES used in this interface are as follows:

- ARM_PH - Used for any Dept, Class, or Subclass calculations where promotions are not involved.
- ARM_PH_PROMO - Used for any calculations that focus on the involvement of a promotion.
- ARM_PH_CS - Used for any calculations that process results for a customer segment.

- ARM_ITEM - Used by user-created runs where the result level is at the Item level. This is not used by batch automation, as Item results are not calculated automatically.
- If there is a need for different set of configuration values for each level of the hierarchy, the HIER_LEVEL_NAME can be provided with the appropriate value to indicate what the configurations are for (SBC, CLS, DEPT).

Science Algorithms/Services

This section describes the science algorithms and services.

ARM_PH

This implementation calculates association rules for a configurable set of product hierarchy levels. It supports the creation of association rules for Sub Classes, Classes, and Department, which can be controlled by a system configuration. All system configurations that affect this algorithm exist in the RSE_CONFIG configuration table, and are named with "ARM_PH_" as the prefix. Because this implementation supports being run for multiple hierarchy levels, if there is a need to set a configuration uniquely for a specific hierarchy level, this can be accomplished via the RSE_CONFIG_CODE table using the hierarchy level name as the PARAM_CODE value. If no such row exists in RSE_CONFIG_CODE, then the configuration will be taken from the corresponding RSE_CONFIG row.

ARM_PH_PROMO

This implementation calculates association rules at the Sub Class level of the hierarchy and is restricted to only rules where the IF side of the rule is promoted and the THEN side of the rule is not promoted. In order to be able to execute this and have results for this implementation, it is necessary to provide promotion details with the sales transaction data. All system configurations that affect this algorithm exist in the RSE_CONFIG configuration table and are named with "ARM_PROMO_" as the prefix.

This implementation is used to focus on how products are associated when that promotion is in effect. This data can help a retailer understand the sales patterns that exist when promotions are involved, which can help the retailer avoid promoting too many items in an effort to help improve profit.

ARM_PH_CS

This implementation calculates association rules at the Sub Class level of the hierarchy and is restricted to rules where the IF side of the rule is promoted and the THEN side of the rule is not promoted. In order to be able to execute this and have results for this implementation, it is necessary to provide promotion details with the sales transaction data. All system configurations that affect this algorithm exist in the RSE_CONFIG configuration table and are named with "ARM_CS_" as a prefix.

This implementation provides the same type of information as the ARM_PH_PROMO implementation; however, it provides results that are specific to a customer segment. Therefore, this implementation requires the receipt of transactions that include the customer ID of the customer who purchased the transaction and the customer segment dimension, along with the association of the customers to each customer segment.

Configurations

There is a consistent pattern in the naming of the configurations for the AA implementation. As described above, each implementation has a specific naming prefix. The suffixes are also similar across the implementations. These suffixes are described in more detail in [Table 8-2](#).

Table 8-2 Implementation Suffixes

Suffix	Example	Description
HIER_LEVEL	SBC	Indicates the name of the hierarchy level that the process is to be executed for. The values here are the same values as provided as LEVEL_NAME values in the W_PROD_CAT_DH interface. Not applicable to ARM_PH.
TOP_LEVEL	SBC	Indicates the highest level of the product hierarchy for which processing should be executed. Can contain SBC, CLS, or DEPT. Only applicable to ARM_PH.
MIN_SUPPORT	.001	Expresses the minimum percentage of transactions that are required to have the set of items in the same transaction.
MIN_SUPPORT_TXN_CNT	1000	In the event that sales volume is low, this is another way to express the minimum number of sales transactions that are required for the set of items to be sold together. The implementation uses the greater of the two values.
MIN_CONFIDENCE	.05	The minimum confidence value as calculated by the rule mining algorithm for an association rule.
MIN_REV_CONFIDENCE	.05	The minimum confidence as calculated by reversing the placement of the numbers in the calculation. Setting this value higher can help prevent redundancy in the rule expressions where the IF and THEN items are transposed.
MIN_LIFT	.05	The minimum lift as calculated by the rule mining algorithm for an association rule.
MAX_LIFT	100	The maximum lift as calculated by the rule mining algorithm for an association rule.
MAX_SET_SIZE	2	The maximum number of hierarchy members to include in the resulting rules. The set size includes the count of both the IF and THEN components. The maximum allowed is four, although it can be an expensive to calculate that many components.
MAX_RULE_COUNT	9999	The maximum number of rules that are retained per execution of the algorithm, per set size. This allows for the reduction of results to eliminate less important results.
WEEK_CNT	1	The number of weeks that are processed when the execution runs. Care should be taken when changing this to more than one week, as this can negatively affect performance.

Data Output

The results of the association rule mining can be obtained from two export interface files. One export interface contains summary information (mba_arm_run_exp) about the execution, along with various metrics that explain what the results are for. The second export interface file

(mba_arm_result_exp) contains the details for each execution of the process. It is possible that a run may not contain any results to be exported. The data between the two interfaces can be joined to each other by the first column in each interface file (the RUN_ID).

In addition to metrics that quantify the rule (its frequency, its confidence, and its lift), the results also include sales values for the different components of the association rule. These sales values can help quantify the involved sales volume that is involved in the association rule.

Even if the weekly process that runs is executed for a single week each time, it is still possible to estimate the effects of the rule across multiple weeks by aggregating data across the weeks. The process for doing this requires locating the same product set across the different weeks within the same execution type. This means to join data in the mba_arm_result_exp interface by if_prod_ext_key1, if_prod_ext_key2, if_prod_ext_key3, if_promo_flg1, if_promo_flg2, if_promo_flg3, then_prod_ext_key, then_promo_flg, and the data in the mba_arm_run_exp interface by run_type, if_hier_level, then_hier_level, loc_ext_key, custseg_ext_key. The data between the mba_arm_run_exp and mba_arm_result_exp files are joined by the run_id.

Once this appropriate data has been gathered, the various sales metrics can be aggregated as needed. In order to calculate a new set of Frequency, Confidence, Lift, or Reverse Confidence values for a rule, it is possible to recalculate the values, as shown below. Note that in these calculations, the following abbreviations are used: run = mba_arm_run_ext, result = mba_arm_result_exp.

$$\text{Frequency} = \text{SUM}(\text{result.rule\_txn\_count}) / \text{SUM}(\text{run.tot\_txn\_cnt})$$
$$\text{Confidence} = \text{SUM}(\text{result.rule\_txn\_cnt}) / \text{SUM}(\text{result.if\_tot\_txn\_count})$$
$$\text{Reverse Confidence} = \text{SUM}(\text{result.rule\_txn\_cnt}) / \text{SUM}(\text{result.then\_tot\_txn\_count})$$
$$\text{Lift} = \text{SUM}(\text{result.rule\_txn\_cnt}) * \text{SUM}(\text{run.tot\_txn\_cnt}) / \text{SUM}(\text{result.if\_tot\_txn\_count}) / \text{SUM}(\text{result.then\_tot\_txn\_count})$$

9

Assortment Recommender

This chapter provides details about the Assortment Recommender, a batch-run system that provides recommendations for changing the current assortments in each store and each category in order to increase revenue or gross profit from that category at that store.

Prerequisites

The Assortment Recommender relies on output from the DT application of the application. In particular, every category/store combination that is to receive assortment recommendations must have DT results in an approved DT version. Some of the concepts used in the Assortment Recommender, such as substitutable and incremental demand, are concepts that the DT application also uses.

Producing Better Assortments

The Assortment Recommender starts from the current assortments for each category in each store and finds assortment changes that will increase either the total revenue or total gross profit of the assortment. Each category-store combination is optimized separately.

The total revenue or total gross profit calculation accounts for

- Cannibalization effects, by using demand transference, and
- Halo effects, meaning additional revenue a SKU in category brings by encouraging the purchase of complementary SKUs in other categories.

The assortment changes recommended are those that improve revenue or gross profit after accounting for the above effects. For example, the Assortment Recommender may recommend dropping an item that is very similar to the other items in the assortment and replacing it with an item that is less similar to the other items, because dropping the similar item does not decrease total revenue by much, and adding the dissimilar item brings in additional revenue. Similarly, the Assortment Recommender may recommend dropping a SKU that brings little halo revenue in favor of a SKU that brings in more halo revenue (or gross profit).

The calculation of revenue or gross profit does not include any revenue that an assortment change in other categories may bring through halo effects. Each category at a store is optimized separately, so when optimizing a category B, it is not possible to include revenue that another category A may bring to B through the halo effect of a SKU in A on B.

The Assortment Recommender lets you choose which of the following to maximize:

- total assortment sales units
- total assortment revenue
- total assortment gross profit

These quantities include halo sales units, halo revenue, or halo gross profit, respectively

The Assortment Recommender runs as a batch process and on a set schedule produces recommended assortment changes for each category/store combination using whatever is the

current assortment at the time it runs. See the following sections for how the schedule can be configured.

Run Groups and Run Frequency

It is unnecessary to obtain frequent assortment-change recommendations for every single category/store combination. For many category/store combinations, recommendations may only be necessary at infrequent intervals, or not at all. The Assortment Recommender provides interfaces to allow external control of the run frequency of category/store combinations in the following way.

The interfaces allow for defining a run group, which consists of a set of categories and a set of stores. When a run group executes, every combination of category/store from the set of categories and set of stores receives new recommendations. The run group as a whole is associated with a run frequency, which specifies how often the category/store combinations in it receive new recommendations. In this way, the run group is a mechanism to control:

- The category/store combinations that receive recommendations. For example, if, for category A, only stores 1, 2, and 3 should receive recommendations, then a run group can be set up with category A, and stores 1, 2, and 3.
- The frequency of recommendations. A retailer may be particularly interested in certain key categories, so it makes sense to schedule recommendations only for those categories. However, not every category may need frequent recommendations.

Keep in mind that when the run group executes, every combination of category/store in the run group receives recommendations. If the run group has 10 categories and 500 stores, this is $10 \times 500 = 5,000$ sets of recommendations and 5,000 separate calculations to produce those recommendations.

The Run Group Parameters

Various parameters control how the recommendation calculation proceeds when a run group executes. These parameters must be present in the run-group tables in the database. Within a run group, each category has its own set of these parameters because the parameters depend on the category. The parameters are:

- The set of Must-Keep SKUs. Each category in a run group can have a list of such SKUs, which indicate to the Assortment Recommender that these SKUs must not ever be removed from the assortment. For example, key items for the retailer in the category must be on the list. The list can be empty, in which case the assortment recommender is free to swap out any SKU currently in the assortment.
- The Assortment-Size Change. This is an integer that can be negative, positive, or 0. It indicates the change the Assortment Recommender must make to the number of SKUs in the assortment. Suppose the value is C. Then the Assortment Recommender will change the assortment size by C. If C is 0, then the assortment size stays the same. If negative, the assortment size decreases by -C, and if positive the assortment size increases by C. There is one value of C for each category in a run group. For a given category, C applies to all of the stores in the run group. It may not be possible for the Assortment Recommender to achieve a change of C for a particular category in a particular store, but it will attempt to get as close as possible.
- The Min-Keep Percent. This is a non-negative percentage. Suppose the value is M. Then the Assortment Recommender keeps at least M percent of the SKUs in the assortment to be the same. Suppose M were 50 percent. Then at least half the assortment will consist of SKUs that are already in the assortment, but the Assortment Recommender is free to choose which 50 percent to change. This parameter, along with the Must-Keep SKUs, is

useful for ensuring that the Assortment Recommender does not make too many changes to the assortment.

- The assortment metric to maximize. The choices here are: sales units, revenue, or gross profit. This is a total over the entire assortment, and the Assortment Recommender recommends assortment changes that increase the chosen metric from what it is for the current assortment. The choice of assortment metric is per category within a run group, and it applies to all of the locations in the run group. For example, if a certain category is a loss leader but drives the customer to make other purchases, then the choice of metric for this category might be sales units.

Data Used by the Assortment Recommender

The Assortment Recommender requires the following data to generate assortment recommendations. Each data element is derived automatically from other data in the application schema and fed into the Assortment Recommender. The following list describes the data and how it is derived.

Required Data

- The current assortment for each category at each store. As discussed above, for each category/store combination in a run group, the Assortment Recommender starts with the current assortment and makes changes to it to increase the chosen assortment metric. For each category/store combination, the system takes the SKUs that were selling in the store during the last available week of historical data.

The weekly sales-units rate of each SKU in the current assortment. This is calculated through an average over the most recent four weeks of historical data.

- The price of each SKU in the current assortment. Historical price data is not used for this, but instead the total revenue over the most recent four weeks of historical data is divided by the total sales units over the same four weeks. This provides an average historical price, based on the last four weeks of historical data.
- The gross profit of each SKU in the current assortment. Historical price or cost data is not used; instead, an average is taken similarly to the price calculation. The total gross profit is taken over the most recent four weeks of historical data and divided by the total sales units over the same four weeks.

The sales-units rate, price, and gross profit are required in order to support the possible assortment metrics.

Notice that in addition to data about the current assortment, the Assortment Recommender requires data about possible SKUs that it can swap into the assortment, since otherwise, in the case of keeping the assortment sizes the same or expanding the assortment, no recommendations would be possible. (The case of decreasing assortment sizes is discussed separately below.) For each category/store combination in a run group, the system calculates the following for the possible SKUs to be swapped into an assortment:

SKU Data

- For a category/store combination, the set of new SKUs is the SKUs that are in the current assortments of other stores but not in the current assortment of this store. (See above for how the current assortment of a store is determined.)
- The price of the new SKU is the average of the prices in the stores in which it is part of the current assortment. The price at each store in which it is selling is determined as described in "[Required Data](#)".

- The gross profit of the new SKU is the average of the gross profits in the stores i in which it is part of the current assortment. The gross profit at each store in which it is selling is determined as described in "[Required Data](#)".
- Assigning the sales units rate of the new SKU is the trickiest to handle, since the sales-units rate of the new SKU must be forecast based on the assumption that it is selling at a store where it may not have sold before. Here, the Assortment Recommender identifies like items of the new SKU among SKUs that are currently in the assortment at the store, and from the like items it makes a forecast of the new SKU's sales-units rate at this store. Identifying the like items is done through the use of similarities. For more information about similarities, see the section "[The Role of Attributes in Calculating Similarities](#)".

If the Assortment-Size Change parameter is set to a negative value, then it is possible to decrease the size of the assortment without having data about possible SKUs to swap into the assortment, as in this situation the Assortment Recommender would simply be finding SKUs to delete from the assortment while still maximizing the chosen assortment metric.

Halo Effects

As mentioned above, the Assortment Recommender accounts for halo effects when calculating the selected assortment metric. This calculation uses the output of the Affinity Analysis. AA determines halo effects at the sub-class level. That is, sub-class A of one category has a halo effect on sub-class B of another category, meaning some significant fraction of the people who purchase in a SKU in A also purchase a SKU in B. Suppose the Assortment Recommender is running for a particular category C. The Assortment Recommender, when it considers putting a SKU C into the assortment, adjusts upward the amount of the assortment metric that C brings in order to include the halo effect. For example, suppose C is a SKU in sub-class A, and sub-class A brings a halo lift of 10 percent to sub-class B. If the metric the Assortment Recommender is maximizing is sales units, then to the sales units U of C itself, the Assortment Recommender adds sales units of $0.1U$ to represent the sales units of B bought by purchasers of C. Similarly, if the chosen metric is revenue, then to the revenue brought by C itself, $0.1U$ times the average price of SKUs in sub-class B is added. The average price of SKUs in sub-class B is calculated by a weighted average of prices, with the weights being the weekly sales-units rates.

The handling of gross profit is similar to the handling of revenue, except that a weighted average of gross profits of B is used instead of the weighted average of prices.

The above discussion involves adding SKU C to the assortment, but the same discussion holds if the Assortment Recommender is removing SKU C from the assortment.

Troubleshooting

Several conditions can prevent the Assortment Recommender from producing recommendations for specific category/store combinations. When any of these occur, the Assortment Recommender will not produce an error but will simply not produce recommendations for the particular category/store combination.

- The DT application was not run for a particular category, or the category does not have an approved DT version associated with it. This means the category does not have results from the DT application, and without those results, it is not possible for the assortment recommender to run since it cannot account for demand-transference effects. In this case, the category will not receive any assortment recommendations regardless of store.
- The Assortment Recommender was not able to find any new SKUs for the particular category/store combination. In the above description about finding the set of new SKUs, it is possible that the procedure does not find any new SKUs at all, perhaps because all of

the stores are assorted identically at that point in time for this particular category. This may happen with categories that are less important to the retailer, so that the retailer does not see any benefit in tailoring the assortment within each store.

- The Assortment Recommender was not able to find any assortment changes that result in an increased assortment metric. This can happen if:
 - There were no new SKUs available (see previous item).
 - The number of new SKUs available was very small.
 - The run-group parameters for the category are too restrictive. For example, too many SKUs are listed as must-keep SKUs or the min-keep percentage was set too high (greater than 80 percent).
- The Assortment Recommender may run, but without using halo effects if the halo effects are not available. For example, the AA may not have run or may not have produced halo effects for the category in question.

10

Size Profiles

This chapter describes the Size Profiles (SP) Cloud Service module.

Overview

Size Profiles (SP) is a module under AIF Profile Sciences and is used to estimate the distribution of demand across different sizes (size profile) for different merchandise and location levels.

Size profile is estimated at different levels of merchandise and location. The lowest level of estimation is style-color (for merchandise) and store/warehouse (for location). Size profiles may also be estimated at higher levels of aggregation on both the merchandise and location dimensions (for example, at subclass-store or style/color-region), depending on the retailer's requirements.

Data Requirements

This section provides information about setting up the data that the SP application uses, including guidelines regarding the expectations for the data element requested and where it is used. Information about these files can be found in Oracle Retail Insights Cloud Service Suite/ Oracle Retail Analytics and Planning Cloud Services Data Interface.

Most of the AI Foundation Cloud Services (AIF) data is pushed in two-step process.

1. First, data is loaded, using CSV and W_ interfaces and jobs in the AIF DATA standalone schedule in POM. Then, RADM_REFRESH_JOB must be run to refresh the table statistics before any AIF Apps job is run (though REFRESH_RADM_JOB is automatically executed as part of most ad hoc processes in AIF Data).
2. Second, appropriate AIF Apps jobs are used to push data into AIF applications. At the time of implementation, the user will only use ADHOC jobs, not any batch jobs. Batch jobs (described in "POM Jobs") are necessary to put the system on a batch schedule.

Note that some of the jobs require relevant configuration parameters to be specified with client-specific values. If incorrect configuration values are used, a job may run without errors, but will not produce the desired data in the target tables. If the client or implementation team wants to view the progress of the job or any errors in order to file an SR, then AIF provides database logging in a table called RSE_LOG_MSG. In addition, analytic logs generated by the programs that run optimization can be found in RSE_ANALYTIC_LOG_MSG. More details are provided in the Troubleshooting section in this chapter.

Hierarchy Data

The three types of hierarchies are Location Hierarchy, Merchandise Hierarchy, and Calendar Hierarchy.

- Location Hierarchy (ORGANIZATION.csv). An example of location hierarchy is: CHAIN ' COUNTRY ' REGION ' DISTRICT ' STORE.

Note that the E-com channel can be defined as part of the location hierarchy. Relevant interfaces are: W_INT_ORG_ATTR_DS, W_INT_ORG_DHS, W_INT_ORG_DS. These are all loaded using the ORGANIZATION.csv file.

- Merchandise Hierarchy (PRODUCT.csv). An example of merchandise hierarchy is as follows: CHAIN ' COMPANY/BANNER 'DIVISION ' DEPARTMENT 'CLASS ' SUBCLASS ' STYLE ' COLOR ' SIZE (SKU). The Size Profiles application is used for fashion apparel. Therefore, it is expected that the extended merchandise hierarchy is provided, that is, Style and Color are provided through the relevant interfaces.

The STYLE, COLOR and SIZE SKUs are expected to be provided in the W_PRODUCT_DS interface, while the levels between CHAIN and Subclass are provided via W_PROD_CAT_DHS interface. In addition, there must be consistency between the levels provided when using an extended hierarchy. W_PRODUCT_ATTR_DS is used to indicate the relationship between Style, Style/Color, and Style/Color/Size for an extended hierarchy. All of these tables will be populated when providing a PRODUCT.csv file as a foundation file input.

Here are the specific fields:

PRODUCT_ATTR13_NAME = PROD_NUM for the Style (for example, 0000190086820900)

PRODUCT_ATTR14_NAME = PROD_NUM for the Style/Color (for example, 190086834203)

PRODUCT_ATTR15_NAME = PROD_NUM for the Style/Color/Size (for example, 1975699).

The value of PROD_NUMs is the same as the value in the W_PRODUCT_DS.PROD_NUM interface.

Here is what this looks like:

Table 10-1 Hierarchy Data

PROD_NUM	PRODUCT_ATTR13_NAME	PRODUCT_ATTR14_NAME	PRODUCT_ATTR15_NAME
STYLE_PROD_NUM	STYLE_PROD_NUM		
STY/ COL_PROD_NUM	STYLE_PROD_NUM	STY/COL_PROD_NUM	
STY/ COL_SIZ_PROD_NUM	STYLE_PROD_NUM	STY/COL_PROD_NUM	STY/ COL_SIZ_PROD_NUM

Note

For Size Profiles Cloud Services, the Extended Product Hierarchy must be loaded. . Prior to loading data into AIF tables, make sure that the configuration LOAD_EXTENDED_PROD_HIER is set to "Y". This can be done using Manage System Configurations (as described in Control and Tactical Center) and then selecting RSE_CONFIG from the drop-down list.

Note

The mapping of columns and the example mentioned above for product hierarchy can be different, depending on the source of the data. For example, if the data is loaded from RMS Cloud, the mapping will be as follows:

- PRODUCT_ATTRPRODUCT_ATTR14_NAME = sku ID (a.k.a. style/color/size)13_NAME = sku parent ID (a.k.a.style)
- PRODUCT_ATTR14_NAME = sku ID (a.k.a. style/color/size)
- PRODUCT_ATTR16_NAME = differentiator ID (for example color, or any differentiator that is used in-between style and sku).
- Prior to loading data into AIF tables, make sure that the configuration EXTENDED_HIERARCHY_SRC is set properly according to the source of data which can be RMS or non-RMS.

- Calendar Hierarchy. This is one of the core hierarchies. The retailer can specify the calendar depending on their business requirements (for example, fiscal calendar). The relevant interface for this is W_MCAL_PERIOD_D, loaded using CALENDAR.csv file.

Sales Data

Historical sales data is required at the sku-store-week level and must be provided through SALES.csv file (W_RTL_SLS_TRX_IT_LC_DY_FS staging table). If the sales transaction data is provided, it will be aggregated by the application to the sku-store-level. It is necessary that the sales transaction data be provided at the lowest level of product hierarchy (that is, the level in RSE_HIER_LEVEL that has LEAF_NODE_FLG = 'Y' for HIER_TYPE_ID = 3).

When the system is in production, the latest incremental sales data is obtained as part of the batch process.

Inventory Data

Two methods are available for generating size profiles. For the Optimization method, historical inventory must be provided at the sku-store-week level. To generate profiles using the Normalize method, inventory data is not required. The Normalized method will not perform most of the data pre-processing stages and will not use a statistical method for estimating profiles. Therefore, the normalized profiles will not be as accurate as optimized profiles specially for categories with many stock-out weeks. Historical weekly inventory data can be specified through INVENTORY.csv

Product Attributes

Product attributes are optional. If provided, they can be used in the escalation of the profiles (which is a means of clustering style-colors and data pooling to achieve more robust size profiles). The SP supports selecting up to three product attribute for the escalation. Product attributes must use W_RTL_ITEM_GRP1_DS since it can support any number of attributes.

Product Images Data

Product images that are available on a customer-hosted web server can be viewed in the Size Profiles UI. The W_RTL_PRODUCT_IMAGE_DS.dat interface contains a column called

PRODUCT_IMAGE_ADDR, which can contain the full URL to an image of the product. This URL must be in the following format:

`http[s]://servername[:port]/location/filename.extension`

For example:

`PRODUCT_IMAGE_NAME = imagename.png`

`PRODUCT_IMAGE_ADDR = http://hostname/url/imagename.png`

`PRODUCT_IMAGE_DESC=` Short description of the image

The Size Profiles application running in the cloud does not directly access these images, so there is no need to expose these images outside of the customer's firewall. As long as the user of the SP application has access to the URL while running the SP application, then the user's web browser will be able to resolve the URL and retrieve the images for display when the user chooses this option. The images must be in a file format that the web browser can display. Since the images shown in the UI are small, these images do not need to be high quality images. The size of the image files will affect the time it takes to render them.

Season

The selected season for the run is used in two ways: The historical data period is set based on the start/end date of the selected season, one year ago. In addition, size profiles are generated for products that are selected for the run *and* are mapped to the selected season.

The season definition and the item-season mapping are required if the user wants to filter the historical data based on a specific historical season. Alternatively, user can select "no season" when setting up the run. In that case, they will be able to specify any start/end date for the historical period and all items will be considered in the data processing as long as they had any sales within those periods.

The season definition can be loaded through SEASON.csv (staging files: W_RTL_SEASON_DS and W_RTL_PHASE_DS). In addition, the W_DOMAIN_MEMBER_LKP_TL should have data for DOMAIN_CODE of SEASON and PHASE. The mapping between items and season is also required and can be loaded through PROD_SEASON.csv (staging file W_RTL_SEASON_PHASE_IT_DS).

Size and Size Range Data

The mapping between SKUs and sizes, as well as the mapping between sizes and size ranges, are both required. This data can be provided through DIFF_GROUP.csv (staging files W_RTL_ITEM_GRP1_DS, W_RTL_DIFF_GRP_DS, and W_RTL_DIFF_GRP_D_TL). It is also necessary to set the value of SPO_DIFF_TYPE in RSE_CONFIG to be the same as the diff_type that is in W_RTL_DIFF_GRP_D for size attribute.

In addition to these two interfaces, it is expected that the column prod_attr_grp_id will be populated in W_RTL_ITEM_GRP1_D for the PROD_GRP_TYPE that is associated with Size.

For the jobs that must be run to load data to above tables, see "Batch and Ad-Hoc Jobs".

Note

It is necessary that all attributes, including the Size attribute, are provided at the lowest level of product hierarchy (that is, the level in RSE_HIER_LEVEL that has LEAF_NODE_FLG = Y).

Configurations

[Table 11-2](#) shows the configurations that must be provided at the time of implementation. These configurations can be viewed and modified using Strategy & Policy Management under Control & Tactical Center. In this dashboard, you navigate to **Manage System Configurations**, select **ALL** for application, **RSE_CONFIG** for table, and filter the table by APPL_CODE=SPO.

Table 10-2 Configurations

PARAM_NAME	PARAM_VALUE	DESCR
DEFAULT_APPL_USER	SPO_BATCH_USER	User identifier to be used for batch activities that require user tracking.
RECENT_RUNS_DAYS_BACK	30	Number of days following creation after which successful runs do not appear in corresponding left-hand-side UI panel.
SPO_DIFF_TYPE	S	Diff Type value to be used for SIZE diff
SPO_DIFF_TYPE_COLOR	C	Diff Type value to be used for COLOR diff
SPO_EXPORT_FETCH_SIZE	1000	Fetch Size for exporting records using SPO User Interface Export To Excel.
SPO_LOC_HIER_PROCESSING_LVL	6	Level Identifier for the level of the Location Hierarchy at which we process Size Profile. Default 6 is to process at STORE Level.
SPO_LOC_HIER_TYPE	2	Hierarchy Type Identifier for the Location Hierarchy Type.
SPO_PROD_HIER_PARENT_LVL	8	Level Identifier for the level of the Product Hierarchy of the SKU-Parent. Default 8 is the STYLE-COLOR Level.
SPO_PROD_HIER_PROCESSING_LVL	9	Level Identifier for the level of the Product Hierarchy at which we process Size Profile. Default 9 is to process at SKU Level.
SPO_PROD_HIER_TYPE	3	Hierarchy Type Identifier for the Product/Merchandise Hierarchy Type.
SPO_PROF_EXP_LOC_LVL	4	Mark USER_BY_AIP=Y Where active Season Size Profiles created at Store level in Export DB view
SPO_PURGE_ALL_RUNS	N	Indicates to clean up all run data.
SPO_PURGE_RETENTION_DAYS	30	Number of days to wait before permanently deleting run data.
SPO_NO_SEASON_START_DATE		Override the start date of historical data when No Season option is selected. If this value is provided, the start date for the No Season record in SPO_SEASON is set to this value. If this is NULL, the start date is driven based on the earliest week available in RSE_CAL_HIER. Default is NULL but if provided, date format should be YYYYMMDD.

Table 10-2 (Cont.) Configurations

PARAM_NAME	PARAM_VALUE	DESCR
SPO_NO_SEASON_END_DATE		Override the end date of historical data when No Season option is selected. If this value is provided, the end date for the No Season record in SPO_SEASON is set to this value. If this is NULL, the end date is driven based on the latest week available in RSE_CAL_HIER. Default is NULL but if provided, date format should be YYYYMMDD.
MIN_SLS_PROF_OVER_CORR_A LERT	0.001	Exclude sizes with low sales contribution from over-correction alert: A value greater than 0 and less than or equal to 1. This is the minimum value of escalation level sales profile that a size should have to be considered in the over correction alert check.
SPO_SALES_TYPE	RPC	Indicates the type of historical sales that goes into all stages of size profile generation. Valid values are RPC (regular+promo+clearance), RP (regular+promo) and R (regular).
SPO_GEN_WH_PROF_FLG	N	A Y/N Flag indicating whether size profiles should be generated for virtual Warehouses.

The configurations that are used by the Size Profiles science algorithm are set when creating a run in the UI. Each configuration has a default value that is defined in SPO_DFLT_PARAMETER. The default values can be viewed and modified using the Manage System Configuration screen. Apart from the algorithm parameters that are visible in the UI and can be set when creating a run, there are few high-level configurations that can be managed only in the Manage System Configuration screen. The following table shows these configurations.

Configurations of the Export File

In addition to the global configurations mentioned above, the merchandise and location levels for export ed profiles must be configured in SPO_EXPORT_CONFIG table. You can modify the configurations in this table using Manage System Configurations (as described in Control and Tactical Center). By default, this table will contain the values shown in the following table. The export file (spo_size_profile.csv) contains approved profiles for all intersections that have current_flg = Y. To disable a certain intersection, you can set the current_flg to N. The export file also has few indicator columns (USED_BY_ALLOCATION, USED_BY_PLM, USED_BY_AIP) that can be used by a downstream application to filter data when reading from the file. To configure an intersection for export without mapping it to any application, you can enter the app_name column as NONE.

Logic of Profiles Spread Down in the Export File

If the configured export level is SKU-parent/location (i.e. style-color/location), no spread down will be applied. The export file will contain style-color/location profiles only if there is a size profile run at the same level and profiles from that run are approved.

If the configured export level is anything other than SKU-parent/location (i.e. style-color/location), approved profiles that are at the configured level, will be exported as-is. If there is no

approved profile that match the configured level,, the export process will find the profile at levels above the configured level and will obtain the profile from closest merchandise and location level.

To generate size profiles for warehouse locations, users should select that option when setting up the run in the UI. When the option is selected for a run, profiles will be generated for all virtual warehouse locations. The input data to the profile estimation for warehouses is the sum of outbound transfer and allocations at style-color/warehouse/week level.

Note

For AIP applications, profiles that are at the configured intersection and for the current active season will have USED_BY_AIP as Y.

Prod Level Name	Loc Level Name	App Name	Current Flg
SBC	LOCATION	AIP	Y
COLOR	LOCATION	ALLOCATION	Y
SBC	LOCATION	ALLOCATION	Y
CLS	LOCATION	ALLOCATION	Y
DEPT	LOCATION	ALLOCATION	Y
COLOR	REGION	PLM	Y

Size Profiles for Warehouse Locations

To generate size profiles for warehouse locations, users should select that option when setting up the run in the UI. When the option is selected for a run, profiles will be generated for all virtual warehouse locations. The input data to the profile estimation for warehouses is the sum of outbound transfer and allocations at style-color/warehouse/week level.

Note

Before creating a run with the option of warehouse profiles for the first time, the job for size range mapping must run so that the mapping for warehouse locations is generated. After the job runs once, the size range mapping for warehouse locations will be available and will just be updated in the subsequent execution of that job. If the daily batch jobs are enabled, the job will run automatically. Alternatively, you can run SPO_MASTER_ADHOC_JOB wit parameter -I.

Merchandise and Location Scope of the Size Profile Runs

When creating the optimization runs in the UI, user will have to specify the merchandise and location nodes for the run. The selection is made in a hierarchical way where for example user will select first the department(s), then class(es), and so on. For higher levels like department, user can select multiple values. For lower levels like style, selecting multiple values is not allowed and only possible value to select is "All". There are few configurations that control the levels that are shown in the UI and the lowest level that multi-select is allowed. These configurations which are listed below should not be changed frequently. If there is a need to change any of these configurations, it is strongly recommended to change them only once (ideally during implementation).

- **MERCH_FILTER_START_LEVEL_ID**: First merchandise level in the merchandise filter drop-down list.
- **MERCH_FILTER_END_LEVEL_ID**: Last merchandise level in the merchandise filter drop-down list.
- **LOC_FILTER_START_LEVEL_ID**: First location level in the location filter drop-down list.
- **LOC_FILTER_END_LEVEL_ID**: Last location level in the location filter drop-down list.
- **MERCH_FILTER_MULTISEL_ALL_LEVEL_ID**: First merchandise level that selecting multiple nodes is not allowed. Only the "All" value can be selected.
- **LOC_FILTER_MULTISEL_ALL_LEVEL_ID**: First location level that selecting multiple nodes is not allowed. Only the "All" value can be selected.

Data Output

The following exports are available under ORASE_WEEKLY_EXTRACT.ZIP. The content of the export files will depend on the source of data (whether it's according to the data format in RMS or not).

Table 10-3 Data Output Exports

Export File Name	Value of EXTENDED_HIERARCHY_SRC in RSE_CONFIG	View Name	Content
spo_size_profile.csv	NON-RMS	SPO_SZ_PROF_EXPORT_VW	Approved Profiles at lowest level (style-color/location) and higher level(s).
spo_custom_export.csv	NON-RMS	SPO_EXPORT_DATA_VIEW	Size profiles at product group level as defined in rse_prod_attr_group.

Batch and Ad-Hoc Jobs

The following batch and ad-hoc jobs are responsible for loading the size, size range, and season data into SPO.

Table 10-4 Data-Loading Batch and Ad-Hoc Jobs

JobName	Description	RmsBatch	ParameterValue	Modules
SPO_MASTER_AD_HOC_JOB	Description Run SPO master script	spo_master.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SIZE_ETL_START_JOB	Start job for size related ETLs	rse_process_state_update.ksh	SPO_SIZE_ETL_PROCESS	SPO_Batch
SPO_SIZE_ETL_SETUP_JOB	Setup job for size ETL	spo_size_setup.ksh	SPO_Interfaces SPO_Interfaces_C NE SPO_Interfaces SPO_Interfaces_C NE	

Table 10-4 (Cont.) Data-Loading Batch and Ad-Hoc Jobs

JobName	Description	RmsBatch	ParameterValue	Modules
SPO_SIZE_ETL_P ROCESS_JOB	Process job for size ETL	spo_size_process.k sh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SIZE_ETL_E ND_JOB	End job for size related	rse_process_state_ update.ksh	SPO_SIZE_ETL_P ROCESS End	SPO_Batch
SPO_PROD_LOC_ SIZE_ETL_START _JOB	Start job for product/location/ size ETLs	rse_process_state_ update.ksh	SPO_PROD_LOC_ SIZE_ETL_PROCE SS Start	SPO_Batch
SPO_SIZE_RANG E_ETL_SETUP_JO B	Setup job for size range ETL	spo_size_range_se tup.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SIZE_RANG E_ETL_PROCESS _JOB	Process job for size range ETL	spo_size_range_pr ocess.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SUB_SIZE_R ANGE_ETL_SETU P_JOB	sub size range ETL	spo_sub_size_rang e_setup.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SUB_SIZE_R ANGE_ETL_PROCE SS_JOB	Process job for sub size range ETL	spo_sub_size_rang e_process.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SIZE_MAP_ ETL_SETUP_JOB	Setup job for size map ETL	spo_size_map_set up.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SIZE_MAP_ ETL_PROCESS_J OB	Process job for size map ETL	spo_size_map_pro cess.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SUB_SIZE_ MAP_ETL_SETUP _JOB	sub size map ETL	spo_sub_size_map _setup.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SUB_SIZE_ MAP_ETL_PROCE SS_JOB	Process job for sub size map ETL	spo_sub_size_map _process.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_PROD_SIZE_ _ETL_SETUP_JOB	Setup job for product/size ETL	spo_prod_size_set up.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_PROD_SIZE_ _ETL_PROCESS_ JOB	Process job for product/size ETL	spo_prod_size_pro cess.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SUB_SIZE_R ANGE_PROD_LO C_ETL_SETUP_J OB	Setup job for sub size/range/product/ location ETL	spo_sub_sz_range _pr_lc_setup.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SUB_SIZE_R ANGE_PROD_LO C_ETL_PROCESS _JOB	Process job for sub size/range/product/ location ETL	spo_sub_sz_range _pr_lc_process.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_PROD_LOC_ SIZE_ETL_END_J OB	End job for product/ location/size ETLs	rse_process_state_ update.ksh	SPO_PROD_LOC_ SIZE_ETL_PROCE SS End	SPO_Batch
SPO_SIZE_ETL_S ETUP_ADHOC_JO B	Ad hoc setup job for size ETL	spo_size_setup.ksh	SPO_Interfaces SPO_Interfaces_C NE	

Table 10-4 (Cont.) Data-Loading Batch and Ad-Hoc Jobs

JobName	Description	RmsBatch	ParameterValue	Modules
SPO_SIZE_ETL_PROCESS_ADHOC_JOB	Ad hoc process job for size ETL	spo_size_process.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SIZE_RANGE_ETL_SETUP_ADHOC_JOB	Ad hoc setup job for size range ETL	spo_size_range_setup.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SIZE_RANGE_ETL_PROCESS_ADHOC_JOB	Ad hoc process job for size range ETL	spo_size_range_process.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SUB_SIZE_RANGE_ETL_SETUP_ADHOC_JOB	Ad hoc setup job for sub size range ETL	spo_sub_size_range_setup.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SUB_SIZE_RANGE_ETL_PROCESS_ADHOC_JOB	Ad hoc process job for sub size range ETL	spo_sub_size_range_process.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SIZE_MAP_ETL_SETUP_ADHOC_JOB	Ad hoc setup job for size map ETL	spo_size_map_setup.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SIZE_MAP_ETL_PROCESS_ADHOC_JOB	Ad hoc process job for size map ETL	spo_size_map_process.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SUB_SIZE_MAP_ETL_SETUP_ADHOC_JOB	Ad hoc setup job for sub size map ETL	spo_sub_size_map_setup.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SUB_SIZE_MAP_ETL_PROCESS_ADHOC_JOB	Ad hoc process job for sub size map ETL	spo_sub_size_map_process.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_PROD_SIZE_ETL_SETUP_ADHOC_JOB	Ad hoc setup job for product/size ETL	spo_prod_size_setup.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_PROD_SIZE_ETL_PROCESS_ADHOC_JOB	Ad hoc process job for product/size ETL	spo_prod_size_process.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SUB_SIZE_RANGE_PROD_LOCATION_ETL_SETUP_ADHOC_JOB	Ad hoc setup job for sub size/range/product/location ETL	spo_sub_sz_range_pr_lc_setup.ksh	SPO_Interfaces SPO_Interfaces_C NE	
SPO_SUB_SIZE_RANGE_PROD_LOCATION_ETL_PROCESS_ADHOC_JOB	Ad hoc process job for sub size/range/product/location ETL	spo_sub_sz_range_pr_lc_process.ksh	SPO_Interfaces SPO_Interfaces_C NE	
RSE_SEASON_PHASE_ITEM_ETL_START_JOB	Start job for season/phase/item ETL	rse_process_state_update.ksh	RSE_SEASON_PHASE_ITEM_ETL_PROCESS Start	SPO_Batch
RSE_SEASON_PHASE_ITEM_ETL_LOAD_JOB	Load job for season/phase/item ETL	rse_season_phase_item_load.ksh		SPO_Interfaces SPO_Interfaces_C NE
RSE_SEASON_PHASE_ITEM_ETL_END_JOB	End job for season/phase/item ETL	rse_process_state_update.ksh	RSE_SEASON_PHASE_ITEM_ETL_PROCESS End	SPO_Batch

Table 10-4 (Cont.) Data-Loading Batch and Ad-Hoc Jobs

JobName	Description	RmsBatch	ParameterValue	Modules
SPO_SEASON_LO AD_START_JOB	SPO_SEASON_LO AD_START_JOB	rse_process_state_ update.ksh	SPO_SEASON_LO AD_PROCESS Start	SPO_Batch
SPO_SEASON_LO AD_JOB	SPO_SEASON_LO AD_JOB	spo_season_load.k sh		SPO_Interfaces SPO_Interfaces_C NE
SPO_SEASON_LO AD_END_JOB	SPO_SEASON_LO AD_END_JOB	rse_process_state_ update.ksh	SPO_SEASON_LO AD_PROCESS End	SPO_Batch

The following jobs are for exporting the outputs.

Table 10-5 Output Exporting Batch and Ad-Hoc Jobs

JobName	Description	RmsBatch	ParameterValue	Modules
SPO_EXPORT_ST ART_JOB	SPO_EXPORT_ST ART_JOB	rse_process_state_ update.ksh	SPO_PROD_SIZE _LOAD_PROCESS Start	SPO_Batch
SPO_EXPORT_PR EP_JOB	SPO_EXPORT_PR EP_JOB	spo_export_prep.ks h		SPO_Export
SPO_SZ_PROF_E XPORT_JOB	SPO_SZ_PROF_E XPORT_JOB	spo_sz_prof_expor t.ksh	#Sys- Opt.SPO_HOME/ data/outfile/ spo_sz_prof_expor t.csv	SPO_Export
SPO_SZ_PROF_E XPORT_ADHOC_J OB	SPO_SZ_PROF_E XPORT_ADHOC_J OB	spo_sz_prof_expor t.ksh	#Sys- Opt.SPO_HOME/ data/outfile/ spo_sz_prof_expor t.csv	SPO_Export
SPO_EXPORT_EN D_JOB	SPO_EXPORT_EN D_JOB	rse_process_state_ update.ksh	SPO_PROD_SIZE _LOAD_PROCESS End	SPO_Batch
SPO_EXPORT_PR EP_ADHOC_JOB	SPO_EXPORT_PR EP_ADHOC_JOB	spo_export_prep.ks h		SPO_Export
SPO_CUSTOM_E XPORT_JOB	Extract customized export data	spo_custom_export .ksh		SPO_Export
SPO_CUSTOM_E XPORT_ADHOC_J OB	Ad hoc job to extract customized export data	spo_custom_export .ksh		SPO_Export

The following jobs can be used to purge runs. The parameters that control this purge process are SPO_PURGE_RETENTION_DAYS and SPO_PURGE_ALL_RUNS in the RSE_CONFIG table. The ad hoc job can take an input that acts as an override to the SPO_PURGE_RETENTION_DAYS value. When SPO_PURGE_ALL_RUNS = Y, then both the batch and ad hoc jobs delete data for all size profile runs, irrespective of the SPO_PURGE_RETENTION_DAYS value.

Table 10-6 Purge Run Jobs

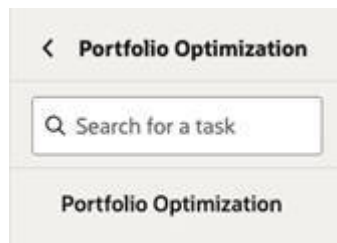
JobName	Description	RMSBatch	ParameterValue	Modules
SPO_PURGE_RUN_START_JOB	Start job for purging SPO run data	rse_process_state_update.ksh	SPO_PURGE_RUN_PROCESS Start	SPO_Batch
SPO_PURGE_RUN_JOB	Purge job for SPO run data	spo_purge_run_data.ksh	SPO_Interfaces SPO_Interfaces_CNE	
SPO_PURGE_RUN_END_JOB	SPO run data	rse_process_state_update.ksh	SPO_PURGE_RUN_PROCESS End	SPO_Batch
SPO_PURGE_RUN_ADHOC_JOB	Ad hoc purge job for SPO run data	spo_purge_run_data.ksh	SPO_Interfaces SPO_Interfaces_CNE	

11

Portfolio Optimization

This chapter describes the Portfolio Optimization (PO) module, where a user can access and manage configurations for optimizing distribution over product areas to align with financial planning for upcoming periods. Depending on the role, a user will see the following link. Typically, the administrative user has access to this link.

Figure 11-1 Portfolio Optimization Link



Configuration

To configure Portfolio Optimization, complete the following steps:

1. Use the Manage System Configurations screen configurations in RSE_CONFIG. For further details, see [Configuration](#). The relevant configurations in RSE_CONFIG that must be reviewed and edited as required are listed in the following table.

Table 11-1 Configuration

APPL CODE	PARAM NAME	PARAM VALUE	DESCR
RSE	PORTFOLIO_LOWEST_PROD_REC_LVL	Default value is 6.	Lowest portfolio product recommendation level

2. Use the Portfolio Optimization screen to set up portfolio scenarios, as follows.
 - a. In Portfolio Optimization, start by setting up a portfolio optimization type. Click the **+** icon in the area at the top of the screen or click the **gear** icon to the far right of the screen title bar. You must create a portfolio optimization type for each sales metric/scope intersection/recommendation intersection combination that is required.
 - b. Once you are done setting up the portfolio optimization types, return to the main Portfolio Optimization screen and review the tiles that are now displayed in the type area at the top of the screen. Click **Start Data Aggregation** in any of the types that have not yet started data aggregation. When data aggregation is complete, the link will change to Data Aggregation Complete. You may see a new link appear below Start Portfolio Aggregation. Click this link if it is present. Once portfolio aggregation completes, the status will change to Portfolio Aggregation Complete. At this point, scenarios can be created and executed to generate recommendations.
 - c. To create a scenario, first click on a configuration type tile at the top of the screen, then expand the Create Scenario section to show the Create form. In the form, you can set

the configurations parameters related to the scenario you want to run. After modifying and reviewing the configuration parameters, click **Create Scenario** (button) to create the scenario. The new scenario will now display in the Scenarios table below. At this point, optimization aggregation and statistics calculations will begin. Once these stages have completed, select the scenario in the table and click **Run** in the Status column. Once the run is complete, the status will change to Complete. Click **Complete** to review recommendations.

- d. On the Results screen, review each of the three recommendations: Highest Growth, Lowest Risk, or Balanced, and approve the one you want to work with by clicking **Approve** and submitting the Approval form. Upon approving a recommendation, the scenario will be approved, and the portfolio aggregation type will be activated. At this point, the recommendation may be imported by MFP or another application that is integrated with Portfolio Optimizations.

Reviewing Optimization Errors

Error logs for scenario optimization aggregation and statistics calculation stages are stored in the RSE_LOG_MSG table. Optimization run execution logs are stored in RSE_ANALYTIC_LOG_MSG. These can be reviewed if the cause of the scenario error status is unclear in the UI.

Example SQL for Optimization aggregation and statistics calculations stages

```
select * from RSE_LOG_MSG where PROGRAM_UNIT like '%PORTFOLIO%' and LOG_LEVEL='ERROR'
order by MSG_TS desc;
```

Example SQL for Optimization Run

```
select * from RSE_ANALYTIC_LOG_MSG where PROCESS_NAME like 'RSE PORTFOLIO RUN JOB';
```

Exporting Portfolio Optimization Results

Export is supported via a web service that a client application such as MFP can call to retrieve approved results.

Service Name: rase/resources/portfolio/v1/MPOptExp

The following table shows the key fields present in the web service request.

Table 11-2 Web Service Request Input Key Mappings

Key Name	Level	Description
requestID	User	Unique Request ID
prodLevel	User	Product Level can be DEPT, CLS, SBC. MFP GA uses DEPT for Merch Target and SBC for Merch Plan.
locLevel	User	Location Level can be COMPANY, PLANNING_CHANNEL, AREA, LOCATION. MFP GA use PLANNING_CHANNEL for Merch Target / Merch Plan.

Table 11-2 (Cont.) Web Service Request Input Key Mappings

Key Name	Level	Description
clndLevel	User	Calendar Level can be YEAR, QTR. Only two are supported in MPO. MFP GA defaults to 'YEAR'.
prodScope	User	Optimization Product Scope level can be CMP,DIV,GRP,DEPT. It must be the level above prodLevel. For GA it will use DIV in MT and DEPT in MP.
locScope	User	Optimization Location Scope level. In initial release, Location Scope is defaulted to same Location level.
keyList	dept/chnl/year	Collection for List of Product/ Location/Calendar key positions for which data is required.
prodKey	dept/chnl/year	Product Position ID (Department ID if Product Level is DEPT)
locKey	dept/chnl/year	Location Position ID (Channel ID if Location Level is PLANNING_CHANNEL).
clndKey	dept/chnl/year	Calendar Position ID (YEAR ID if Calendar Level is YEAR).

The following table shows the key fields present in the web service response.

Table 11-3 Web Service Response

Key/Parameter	Required	Intersection	Comments
MPOptExpHeader	Yes	User	Collection for the Main Request Response Header
responseCode	Yes	User	Example: 100 - For Successful Processing of Data 200 - For Any Key Syntactical Failures 300 - For Any Header level Validation Failures 400 - For Any Key level Validation Failures 500 - Warning for Any Keys Missing Optimization Results
responseCodeMessage	Yes	User	Detailed Response Message for Request
MPOptExpDetail	Yes	User	Collection for the Main Request Response Details

Table 11-3 (Cont.) Web Service Response

Key/Parameter	Required	Intersection	Comments
requestID	Yes	User	Same requestID provided in the input request
keyList	Yes	User	Collection for List of Products/Store Cluster
prodKey	Yes	dept/chnl/year	Same Product ID provided in the input request
locKey	Yes	dept/chnl/year	Same Location ID provided in the input request
cldnKey	Yes	dept/chnl/year	Same Calendar key provided in the input request
optNSIsAmt	Yes	dept/chnl/year	Optimized Net Sales Amount
optGSIsAmt	Yes	dept/chnl/year	Optimized Gross Sales Amount
keyRespCode	Yes	dept/chnl/year	Optional Response Code at Key Level 400 - Requested Key List not Found 500 - Optimization Results not available
keyRespMsg	Yes	dept/chnl/year	Optional Response Message at Key Level.

Data Requirements for Portfolio Optimization

Portfolio Optimization requires at least three years of quality sales transactions to perform meaningful growth calculations. For each year of sales history, the number of weeks with positive sales per year must be at least 5% of all possible sales weeks in the year.

Pre-Pack Optimization

This chapter describes the Pre-Pack configuration module in AI Foundation Cloud Service.

Overview

Pre-Pack Optimization recommends optimal Pre-Pack configurations that meet a store-specific requirement while maximizing supply chain efficiencies. It considers vendor constraints, size min/max requirements, and other common pre-pack related constraints with scenario modeling support. Users can learn about the business impact on changes to constraints. The approval process enables users to review and finalize pre-pack configurations with information on benefits from recommended pre-pack configurations.

Data Requirements

This section provides information about setting up the data that the Pre-Pack Optimization uses, including guidelines regarding the expectations for the data element requested and where it is used. Information about these files can be found in Oracle Retail Insights Cloud Service Suite.

For more details on file specifications, see the AI Foundation Interfaces Guide on My Oracle Support (MOS): Oracle Retail Insights Cloud Service Suite, AI Foundation Cloud Services, Retail Data Extractor for Cloud, and Retail Home Documentation Library at [Doc ID 2539848.1](#).

Hierarchy Data

The three types of hierarchies required are Location Hierarchy, Merchandise Hierarchy, and Calendar Hierarchy. Data is first loaded into AIF Data and then loaded into AIF applications. For additional details, refer to the Data Loads and Initial Batch Processing section of the *Oracle Retail Analytics and Planning Implementation Guide*. If Pre-Pack Optimization is the first module of the AIF applications that is being implemented, then set the following configuration settings in RSE_CONFIG file.

APPL CODE	PARAM NAME	PARAM VALUE	DESCR
RSE	EXTENDED_HIERARCHY_SRC	RMS	Data source providing extended hierarchy data RMS/NON-RMS.
RSE	LOAD_EXTENDED_PRODUCT_HIER	Y	Y/N value. This parameter is used by the product hierarchy ETL to determine if the extended product hierarchy is required.

Sales Data

Historical sales data is required at the sku-store-week level and must be provided. Refer to the Data Loads and Initial Batch Processing section of the *Oracle Retail Analytics and Planning Implementation Guide*.

Size and Size Range Data

The mapping between SKUs and sizes, as well as the mapping between sizes and size ranges, are both required. This data can be provided through DIFF_GROUP.csv (staging files W_RTL_DIFF_GRP_DS, and W_RTL_DIFF_GRP_D_TL). It is also necessary to set the value of SPO_DIFF_TYPE in RSE_CONFIG to be the same as the diff_type that is in W_RTL_DIFF_GRP_D for size attribute.

In addition to these two interfaces, it is expected that the column prod_attr_grp_id will be populated in W_RTL_ITEM_GRP1_D for the PROD_GRP_TYPE that is associated with Size. For the jobs that must be run to load data to above tables, see the Batch and Ad-Hoc Jobs section of this chapter.

Note

It is necessary that all attributes, including the Size attribute, are provided at the lowest level of product hierarchy (that is, the level in RSE_HIER_LEVEL that has LEAF_NODE_FLG = Y).

Supplier Data (Optional)

If supplier data is provided, it is possible to generate Pre-Pack configurations that are supplier-specific. Supplier data can be provided via PRODUCT.csv file. This in turn loads tables W_PARTY_ORG_D and W_RTL_IT_SUPPLIER_D. These two RI tables are used to load RSE_SUPPLIER and RSE_SUPPLIER_ITEM.

External Assortment Plan Data (Optional)

It is possible to load external Assortment Plans and use this as a basis for optimizing pre-pack configurations. Assortment Plan data is expected at the Style Color/Store/Week level. The format of the external plan data is consistent with the Assortment Plan (PLAN5) interface in Retail Insights Cloud Service. The required interface is W_RTL_PLAN5_PROD5_LC5_T5_FS. See the following section to make sure the data level is set correctly.

Interface Name: W_RTL_PLAN5_PROD5_LC5_T5_FS.dat

Setting Data Levels

C_ODI_PARAM needs to be updated before loading any style-color data via the Control and Tactical Center. In the C_OCI_Param table:

- find param_name = 'RI_PLAN5_PROD_LEVEL' and set param_value = 'ITEM_LEVEL1'.
- find param_name = 'RI_PLAN5_ATTR_LEVEL' and set param_value = 'C'

Refer to the section on Planning and Flex Fact Configuration in the *Oracle Retail Insights Cloud Service Implementation Guide* for more details on making updates to the C_ODI_PARAM_VW table.

PLAN data files are picked up by nightly POM jobs. In addition, it is also possible to load them as ad hoc processes via DAT_REPROCESS_ADHOC. Refer to the AI Foundation Data Standalone Process Flows section in the *Oracle Retail AI Foundation Cloud Services Operations Guide*.

Sample Data

W_RTL_PLAN5_PROD5_LC5_T5_FS.dat:

```
PROD_DH_NUM,PROD_DH_ATTR,ORG_DH_NUM,SUPPLIER_NUM,CAL_DATE,PLANNING_TYPE_CODE,VERSION_NUM,
SLS_QTY,SLS_RTL_AMT,SLS_COST_AMT,EOH_COST_AMT,EOH_RTL_AMT,EOH_QTY,INVR_COST_AMT,INVR_RT
L_AMT,INVR_QTY,DOC_CURR_CODE,DATASOURCE_NUM_ID,ETL_THREAD_VAL,INTEGRATION_ID,LOC_CURR_CO
DE
990874190,multi,2041,-1,20230923,NA,0,17.474183200394,2098.0299530574,1070.6304425932,
43.538242705932,85.318456926443,.71060489128625,0,0,0,USD,1,1,990874190~2041~23-
SEP-23~-1~multi~NA~0,USD
1102018,yellow,2154,-1,20230826,NA,0,3.5946690094215,508.60971814305,273.05105795566,
46.936181947753,87.427598522743,.61790655539433,0,0,0,USD,1,1,1102018~2154~26-
AUG-23~-1~yellow~NA~0,USD
1102019,green,2116,-1,20230826,NA,0,4.9103104966421,844.52430231748,511.50704443521,
53.458223625021,88.262262467769,.51318252495941,0,0,0,USD,1,1,1102019~2116~26-
AUG-23~-1~green~NA~0,USD
```

W_RTL_PLAN5_PROD5_LC5_T5_FS.dat.ctx:

```
#TABLE#W_RTL_PLAN5_PROD5_LC5_T5_FS#
#DELIMITER#,#
#DATEFORMAT#YYYYMMDD#
#REJECTLIMIT#1#
#RECORDDELIMITER#\n#
#IGNOREBLANKLINES#false#
#SKIPHEADERS#1#
#TRIMSPACES#rtrim#
#TRUNCATECOL#false#
#COLUMNLIST#
PROD_DH_NUM
PROD_DH_ATTR
ORG_DH_NUM
SUPPLIER_NUM
CAL_DATE
PLANNING_TYPE_CODE
VERSION_NUM
SLS_QTY
SLS_RTL_AMT
SLS_COST_AMT
EOH_COST_AMT
EOH_RTL_AMT
EOH_QTY
INVR_COST_AMT
INVR_RTL_AMT
INVR_QTY
DOC_CURR_CODE
DATASOURCE_NUM_ID
ETL_THREAD_VAL
INTEGRATION_ID
LOC_CURR_CODE
```

Configurations

The following table shows the configurations that must be provided at the time of implementation. These configurations can be viewed and modified using Strategy & Policy Management under Control & Tactical Center. In this dashboard, you navigate to Manage System Configurations, select ALL for application, RSE_CONFIG for table, and filter the table by APPL_CODE=PPO.

PARAM_NAME	PARAM_VALUE	DESC
PPO_CAL_HIER_PROCESSING_LVL	4	Level Identifier for the level of the Calendar Hierarchy at which we process pre-pack/size data. Default 4 is to process at Week Level.
PPO_CAL_HIER_TYPE	11	Hierarchy Type Identifier for the Calendar Hierarchy Type.
PPO_LOC_HIER_PROCESSING_LVL	6	Level Identifier for the level of the Location Hierarchy at which we process pre-pack/size data. Default 6 is to process at STORE Level.
PPO_LOC_HIER_TYPE	2	Hierarchy Type Identifier for the Location Hierarchy Type.
PPO_MIN_STORE_SIZE_COUNT	2	Minimum number of sizes at a store within a style-color.
PPO_PROD_HIER_PARENT_LEVEL	8	Level Identifier for the level of the Product Hierarchy of the SKU-Parent. Default 8 is the STYLE-COLOR Level.
PPO_PROD_HIER_PROCESSING_LVL	9	Level Identifier for the level of the Product Hierarchy at which we process pre-pack/size data. Default 9 is to process at SKU Level.
PPO_PROD_HIER_TYPE	3	Hierarchy Type Identifier for the Product/Merchandise Hierarchy Type.
PPO_SIZE_PROFILE_LOC_LEVEL	6	Level Identifier for the level of the Location Hierarchy at which we process Size Profile. Default 6 is to process at STORE Level.
PPO_SIZE_PROFILE_MERCH_LEVEL	6	Level Identifier for the level of the Product Hierarchy at which we process Size Profile. Default 6 is to process at SUBCLASS Level.
PPO_SIZE_PROFILE_MERCH_LOWEST_LEVEL	9	Level Identifier for the lowest level of the Product Hierarchy for Size Profile. Default 6 is to process at SKU Level.
PPO_SIZE_PROFILE_MERCH_LOWEST_PARENT_LEVEL	8	Level Identifier for the parent level of the lowest level of the Product Hierarchy for Size Profile. Default 8 is to process at STYLE-COLOR Level.

Batch and Ad-Hoc Jobs

In addition to the standard jobs to load data into AIF applications for standard hierarchies and sales data, the following jobs must be run to populate the size, size range, and sub size range information that is required. Note that these jobs must be run even if SPO is not being implemented.

- SPO_SIZE_ETL_START_JOB
- SPO_SIZE_ETL_SETUP_JOB
- SPO_SIZE_ETL_PROCESS_JOB
- SPO_SIZE_ETL_END_JOB
- SPO_PROD_LOC_SIZE_ETL_START_JOB
- SPO_SIZE_RANGE_ETL_SETUP_JOB
- SPO_SIZE_RANGE_ETL_PROCESS_JOB
- SPO_SUB_SIZE_RANGE_ETL_SETUP_JOB
- SPO_SUB_SIZE_RANGE_ETL_PROCESS_JOB
- SPO_SIZE_MAP_ETL_SETUP_JOB
- SPO_SIZE_MAP_ETL_PROCESS_JOB
- SPO_SUB_SIZE_MAP_ETL_SETUP_JOB
- SPO_SUB_SIZE_MAP_ETL_PROCESS_JOB

Refer to [Data-Loading Batch and Ad-Hoc Jobs](#) for specific details.

LifeCycle Pricing Optimization

This chapter describes the Lifecycle Pricing Optimization (LPO).

Overview

LPO is used to determine the optimal pricing recommendations for promotions, markdowns or targeted offers. Promotions and markdowns are at the location level or a price zone. Targeted Offers can be specific to each customer and not just to a location. Pricing recommendations contain answers to the following questions: Which items? When (timing)? and How deep and Who (segments)? It caters to the retailers who are interested in only promotions and markdowns as well as targeted offers that are specific to each customer segment. In order to use targeted offers, the retailer must provide customer-linked sales transactions data; to use promotions and markdowns, the retailer does not necessarily have to provide customer-linked sales transaction data.

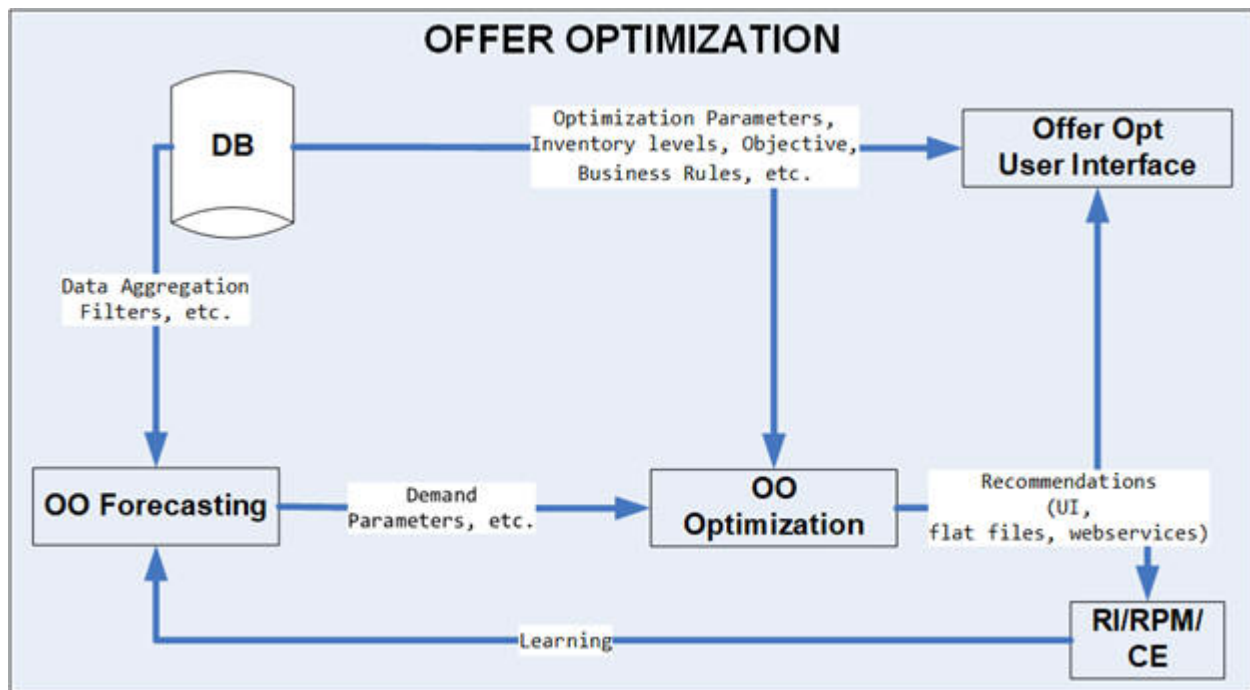
Certain aspects of promotions, markdowns, and targeted offers are important levers for managing the inventory over the life cycle of the product. The application helps in the following:

- Bring inventory to the desired level, not just during the full-price selling period.
- Maximize the total gross margin amount over the entire product life cycle.
- Assess in-season performance.
- Provide updated recommendations each week. This facilitates decision-making that is based on recent data, including new sales, inventory, price levels, planned promotions, and other relevant data.
- Provide targeted price recommendations at the segment-level.

[Figure 13-1](#) shows the conceptual flow of different components in LPO. Oracle Retail AI Foundation Cloud Service and Retail Insights Cloud Service is the core data foundation layer that consumes the retailer data. The forecasting analyzes and mines historical data (along with other data sources) using machine learning algorithms for developing a predictive model to provide the forecasting inputs to the Optimization Algorithm. The Optimization Algorithm obtains inputs such as objectives, budgets, and business rules from the UI, along with optimization parameters from AIF and the data warehouse. The algorithm analyzes the feasible price paths efficiently and generates price recommendations. These recommendations can be viewed in the LPO UI or can be exported to the price execution systems such as Price Management (PM) or Customer Engagement (CE). In addition, a feedback loop from the price execution systems can help the Forecasting component to determine how the offers are performing and adjust the next set of offers based on the sales performance or the response rate. LPO supports two kinds of runs, called "ad hoc runs" and "batch runs." Batch runs are scheduled to run automatically at regular intervals (for example, every week). Each batch run, using latest sales data and inventory levels, updates the parameters and budgets and produces the price recommendations. An analyst can review the results and further accept, reject, or override the price recommendations for each item. Once the analyst finishes the review, the item's recommendation status can be changed to Reviewed. A reviewed recommendation is sent to the buyer for approval. If the user with the buyer role likes the recommendations, that user can submit or approve the recommendation. At this point, the recommendation status is changed to Submitted or Approved. This indicates that the price

recommendations will be sent to an export interface as well as to a price execution system such as RPM with the respective statuses.

Figure 13-1 Lifecycle Pricing Optimization



An optimization can be carried out at the configured processing (or run) location or the price-zone, merchandise level, and calendar level. The user can configure the optimization to use either the price-zone or a node in the location hierarchy (but not both in the same instance). Once the optimization is complete, the recommendations can be generated at a lower level than the processing level (called recommendation levels for merchandise level and location or price-zone level). The location and merchandise level can be any level in the location hierarchy and merchandise hierarchy, respectively. Alternatively, price-zone can be used to define a set of stores (and/or online locations) and items. An example of a price zone is women's apparel in all university-based stores grouped into one price-zone. The usual levels for the run are Region or Price-zone, Department, and Week, and for the recommendation, the levels are Region or Price-zone and Style/Color or SKU (Style/Color/Size). If Targeted Offers is available, then the recommendations will also be generated at the Customer-Segment level, along with location or price-zone and merchandise level. The Promotions and Markdowns are always at the location or price-zone and merchandise level.

The optimization is set up as follows:

- Optimization is done at the run's merchandise and location setup levels. For example, if the run merchandise and location levels are Department and Region, then each optimization job is at the Department-Region level.
- Inventory is rolled to the desired recommendation level for merchandise. Further, the inventory is aggregated across all the locations to the run's location level.
- Price recommendations are generated at the configured recommendation levels for merchandise, location or price-zone, and calendar level and the customer segment level.

For example, if the run location level is Price-zone, the run merchandise level is Department, the run calendar level is Week, and the recommendation level for

merchandise is Style/Color, then the promotion and markdown recommendations are generated at Price-zone, Style/Color, Week and Price-zone, Style/Color, Week and Customer Segment for Targeted Offers.

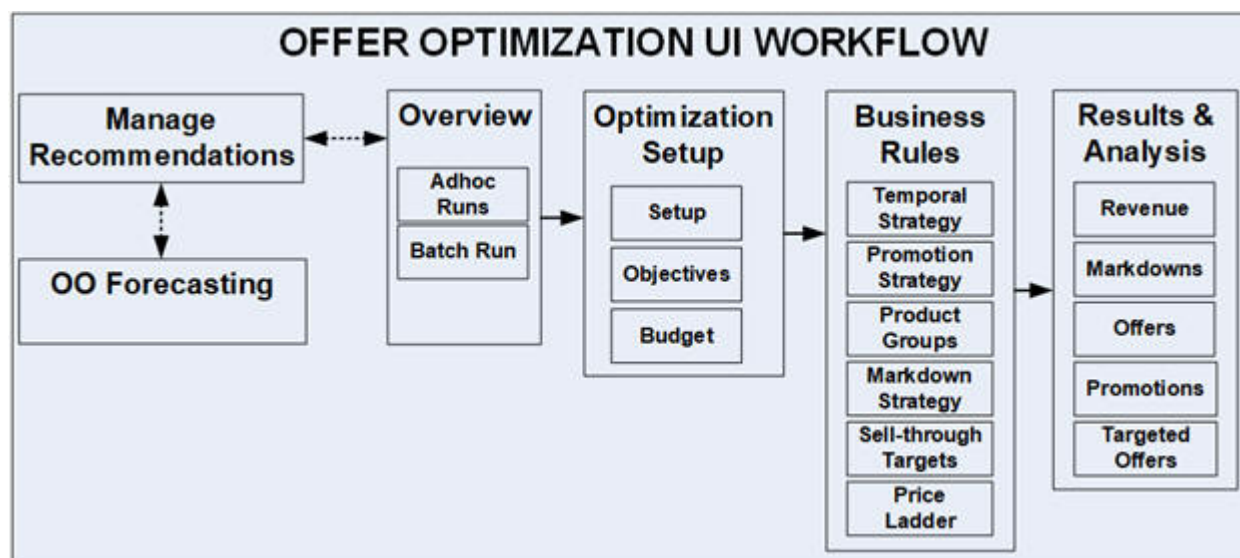
Figure 13-2 shows an overview of the LPO UI workflow, which consists of the following:

- **Overview.** This is the dashboard for the LPO runs. In this tab, you can see a list of all existing runs, along with details that describe each run. The list includes runs created by other users, which you can open in read-only mode. You can create a run, copy a run, open a run, or delete a run. The runs overview has three components, Search, Run Status Tiles, and the Table of Runs.

You can click an existing run or create a new run. Each run is opened in a run tab. The title of the run tab displays "LPO: <Run ID>". This is the main tab where you can specify the business rules and goals for the run. It provides a series of three stages that you progress through in order to set up, run, and analyze the results of the optimization run.

- **Setup.** Used to pick a season, location or price zone and department. It is also used to select the objectives and specify the budgets.
- **Rules.** Used to view or change business rules.
- **Results.** Used to view results and accept, reject, or override recommendations, and recalculate.
- **Forecasting.** These screens provide the user with the ability to review the way in which different components of forecast (for example, baseline, seasonality) affect the price recommendation for an item.
- **Manage Recommendations.** This screen is used to accept, reject, or override and recalculate recommendations. It also allows the user to review, submit, or approve recommendations so that they can be sent to a price execution system. In production, this screen is the starting point and is used most regularly, since the user can send recommendations from batch runs for execution or visit forecasting screens to review how item forecast looks, and finally, decide whether to create a new ad hoc run.

Figure 13-2 Lifecycle Pricing Optimization UI Workflow



The goal of the implementation is to set up and configure an instance to generate optimal price recommendations that satisfy the retailer's business requirements. The implementation

configures the application so that the batch runs complete successfully in a timely fashion and produce valid promotions, markdowns, and forecast recommendations that meet the retailer's requirements. The main implementation tasks involve configuring the following:

- The roles and permissions assigned to users.
- The loading of retailer data.
- The configuration parameters.
- The demand parameters, such as seasonality and price effects, that are used to determine optimal promotion, markdown, and targeted recommendations.
- The business rules that determine constraints that the application takes into account during the optimization process.

Project Planning

This section provides a high-level project planning for implementation by different buckets: functional, data interfaces, forecasting and optimization.

Functional

- Understand the existing business requirements for pricing; and proposed process changes with LPO.
- Verify the functionality that the customer would like to leverage from LPO and identify whether the customer can provide the necessary data.
- Review and identify the business rules (and strategy sets) with the customer.
- Identify the desired flex facts that the customer would like to see in the Manage Recommendation Screen.
- Identify the desired reports for the customer.
- Log and revisit any open requirements.
- Users and user roles, data security/filtering.

Data Interfaces

- Overview. Confirm the list of LPO interfaces to be loaded for the customer
- Foundational Data. Decide on the business date and load the Calendar data. Plan for partitions.
- Foundational Data. Decide on the merchandise and location hierarchy levels; map it to the levels in the LPO hierarchies.
- Foundation Data. Identify the fields in each interface and identify or map to a source in the customer's systems
- Foundational Data. Convert history for sales transactions
- Foundational Data. Convert history for inventory positions.
- Foundational Data. Convert other positional data (costs, prices)
- Foundational Data. Decide on whether to use Price Zones and load the price zone data.
- Validation. Load a small sample of data (for example, 1 month) to verify data is flowing in correctly. Use Innovation Workbench or Data Visualizer for verifications.

- Full historical load. Plan for full load of history.
- LPO Interfaces.

Forecasting Parameters

- Setup. Create a forecast run type and relevant levels for forecast or recommendations
- Setup. Create a forecast run(s) with as many configurations as needed.
- Data Aggregation. Verify that the data aggregation is completed.
- Parameters. Review the forecasting parameters.
- Validation. Assess the forecast quality of the parameters and iterate on forecast runs as needed. Prepare comparison reports using Data Visualizer.

Optimization

- Setup. Configure all the relevant parameters (for example, warehouse virtual allocation algorithm).
- Setup. Verify the business rules and strategies.
- Ad hoc run. Optimize an ad hoc run for optimization for selected merchandise or locations.
- Validation. Review the recommendations and for the quality of the business rules and forecast parameters. Revisit the business rules and forecasting as necessary.
- Ad hoc batch job. Execute an ad hoc batch job to verify the runs are executed successfully for all merchandise/locations.

Walkthrough

Most of the AIF Applications foundation data is pushed in two-step process. Any additional LPO-specific data is directly pushed into the AIF Apps tables. First, data is loaded using CSV and W_ interfaces via AIF DATA schedule in POM. Then, RADM_REFRESH_JOB must be run to refresh the table stats before any AIF job is run. Second, appropriate AIF jobs are used to push data into AIF application tables. At the time of implementation, the user will only use ADHOC jobs, not any batch jobs. Batch jobs (described in "[POM Jobs](#)") are necessary to put the system on a batch schedule. Third, LPO-specific data (that is not available in AIF DATA interfaces) is directly pushed into the AIF applications.

The following sections assumes that the user has obtained access to POM. The POM UI URL is something like this: <host>/POMJetUI. If the user cannot access the POM UI, contact the administrator to obtain the relevant access/user roles. In addition, the user must have access to Innovation Workbench and/or Data Visualizer in order to query or visualize the data in order to verify that the data loaded matches the desired expectations.

Another point to note is that some of the jobs require relevant configuration parameters to be specified with client-specific values. If incorrect configuration values are used, a job may run without errors, but will not produce the desired data in the target tables. If the client or implementation team wants to view the progress of the job or any errors in order to file an SR, then AIF applications provide database logging in a table called RSE_LOG_MSG. Logging can be enabled only with a service request from the client/implementation partner to the support team.

Although this walk-through provides all the necessary steps, the user may want to refer to the relevant section in this guide for more details (for example, the specific fields in an interface).

RAP foundation data (CSV and W_ interfaces)

These interfaces provide the foundation data for the AI Foundation. Execute and verify that all the RAP foundation data has been loaded. For details, refer to the Retail Analytics and Planning Implementation Guide.

For LPO, this includes:

1. **Product Hierarchy.** When Style, Style Color must be loaded, appropriate columns must be populated.
2. **Location Hierarchy.**
3. **Price Zone Groups and Price Zones.** It is not required to provide price zones for LPO, but if the user intends to do pricing by price zones, then these files are required. Price Zones are clusters of stores and Price Zone Groups allow the user to map a merchandise node or set of items to a Price Zone Group.
4. **Sales.** It is best to load at least 20 months of the data to obtain a good signal in the model training.
5. **Inventory.** It is a critical data element for LPO, for, without inventory, the LPO runs would fail.
6. **Inventory Receipts.** This data is used to identify when the item started selling. This data must be provided for LPO.
7. **Customer Segments.** Even when the retailer is not planning to load or use customer segments, a dummy record is required. The AIF ad hoc jobs will create a dummy record when the source data is not provided, and it is not required for the client or implementer to provide these interface if they are not going to use them: W_PARTY_PER_DS and W_RTL_CUSTSEG_DS.
8. **Price and Cost.** This data must contain the price and cost for every product and location combination that had a sale in the sales history. At least a record must be provided for TRAN_TYPE 0 for every product and location combination so that LPO can correctly identify the ticket price, the original price for the item, and the location. Note that this is the minimum required for this data. However, it is highly recommended that the entire price history for that item and location be provided. Otherwise, LPO will incorrectly capture the ticket price changes or regular price changes. It is understandable that some items from the beginning of the sales history will not have the entire price history.
9. **Security.** RAF files must be provided when the user wants to impose security or data filters for the LPO screens based on the merchandise and location nodes. The Manage LPO Recommendation user filter will not show any location or merchandise nodes that are not allowed for a particular user.
10. **Product Images.** The URLs for rendering product images in LPO screens can be provided.
11. **Pricing Groups.** If the user defines pricing groups such as Same Markdown Discount and so on, then they must be provided here for the batch LPO runs to pick up.
12. **Flex Facts (or Custom Columns).** If the user wants to see custom columns in Manage LPO Recommendation, then these interfaces must be provided.
13. Before pushing any data into the AIF application, it is critical to make sure RAP foundation data is accurate and verified. Verification can be done through Data Visualizer or through the Innovation Workbench. Re-loading the RAP foundation data elements is non-trivial and becomes more difficult if you distribute that data from the input data warehouse tables to other downstream systems before validating it.

AIF applications foundation data (RSE_% tables)

Once the interface configuration is complete and the RAP Foundation data has been verified, the user can push the data from RAP into AIF applications. As mentioned before, once database logging has been enabled, users can check for any errors and the progress of the process in RSE_LOG_MSG. Before running the RSE_MASTER_ADHOC with appropriate flags to execute the following jobs, the user must set the correct values for configuration parameters in RSE_CONFIG. This can be done using Manage System Configurations (as described in [Control and Tactical Center](#)) and then selecting RSE_CONFIG from the drop-down list.

1. **Product Hierarchy.** The target table to check is RSE_PROD_HIER. When the user wants to load up to 9-level hierarchy (for example, fashion items with SBC 'STYLE' STYLE/COLOR 'STYLE/COLOR/SIZE'), then two configurations must be verified: LOAD_EXTENDED_PROD_HIER and PROD_ITEMUDA_SRC.
2. **Location Hierarchy.** The target table to check is RSE_LOC_HIER.
3. **Price Zone Groups and Price Zones.** The target tables to check are RSE_PRICE_ZONE, RSE_PRICE_ZONE_GRP, RSE_PRICE_ZONE_LOC, and RSE_PRICE_ZONE_PROD. The user must set the correct configuration level for the merchandise level that is mapped to a PRICE_ZONE_GROUP using this configuration in RSE_CONFIG: RSE_PZG_PROD_LEVEL.
4. **Sales.** The target table to check is RSE_SLS_TXN. This data will be further aggregated into multiple tables at different aggregate levels once the relevant aggregation jobs are executed. For LPO, this table must be populated: RSE_SLS_PR_LC_CS_WK. Returns aggregation will not be required if the user does not plan on using the returns functionality.
5. **Inventory.** The target table to check for weekly inventory position at leaf node levels is RSE_INV_PR_LC_WK_A. Note that the warehouse inventory positions are also loaded into this target table as warehouse is treated like a leaf node on the location hierarchy. The configurations in RSE_CONFIG to check are INV_SRC_RI, RSE_INV_QTY_BOH_FLG, RSE_INV_QTY_IN_TRANSIT_FLG, RSE_INV_QTY_ON_ORD_FLG, RSE_WHSE_INV_QTY_BOH_FLG, RSE_WHSE_INV_QTY_IN_TRANSIT_FLG, and RSE_WHSE_INV_QTY_ON_ORD_FLG.
6. **Inventory Receipts.** The target table to check for inventory receipts at the leaf node level is RSE_INV_HIST_PR_LC_WK. The configuration to check is INV_RC_DT_SRC_RI.
7. **Customer Segments.** The target table to check is RSE_CUSTSEG_HIER. It is required to execute the job to load customer segments (even when there is just a dummy segment).
8. **Price and Cost.** Before pushing the price cost data into AIF applications, the user must make sure that the weekly sales aggregation job has been executed and that the customer segments have been executed. The target table to check at leaf node levels is: RSE_PRICOST_PR_LC_WK.
9. **Security.** Target tables to check are RAF_FILTER_GROUP_MERCH, RAF_FILTER_GROUP_ORG, RAF_SEC_GROUP, RAF_SEC_USER, and RAF_SEC_USER_GROUP.

Direct loads to AIF applications

Some foundation data elements are loaded directly to AIF applications. Here is a description of some of the data elements that are used in LPO.

These files must be prepared into ORASE_WEEKLY_ADHOC.zip.

1. Prepare the relevant _stg.txt file. (Note that some large files will be compressed into _stg.txt.gz files.)

2. Zip all the staging files into an archive named ORASE_WEEKLY_ADHOC.zip.
3. Upload ORASE_WEEKLY_ADHOC.zip to Object Storage using the File Transfer Service (FTS).

Then run the RSE_MASTER_ADHOC with the appropriate parameter. See: <https://docs.oracle.com/en/industries/retail/retail-insights-cloud/21.0/rapog/mstr-dat-load-common-dsgn-ovrvw.htm>

1. **Holidays.** The target table to check is RSE_HOLIDAY. This is a direct load to AIF applications and is loaded through RSE_HOLIDAY_STG. Any validation errors during loading are provided in the RSE_HOLIDAY_BAD table. As mentioned before, the bad tables can be viewed using Innovation Workbench.
2. **Warehouse Allocation Percentages.** RSE_INV_WHSE_LC_PR_ALLOC_STG. The target table to check is RSE_INV_WHSE_LC_PR_ALLOC. This is a direct load to AIF applications. The configuration to check is RSE_PRO_WHSE_ALLOC_FLG, which must be set to N when the client is loading allocation percentages.
3. **Product Attributes.** Product attributes are not required for LPO; however, if the user wants to use them in the Manage OO Recommendation screen, then they must be provided directly to AIF applications. The target tables to check are RSE_PROD_ATTR_GRP_VALUE and RSE_PROD_ATTR_VALUE_XREF.

Forecasting

The next step is to create a forecast run type and run the model training to estimate the demand parameters. These steps are important to determine the appropriate aggregation levels for the next bucket, Optimization.

1. Create the forecast run type at the desired recommendation levels for the offers. For example, the recommendation levels can be STYLE COLOR, PRICE ZONE, and WEEK. Once the correct run type has been created, then user can click **Start Data Aggregation**. The configurations to check are RSE_INV_WHSE_ACTIVITY_USE_FLG (this flag must be set to N as otherwise the aggregation time would increase dramatically. The user can turn this flag on when the batches have started), PMO_PROD_HIER_TYPE (for example, this would be 3 for using extended hierarchy), and PMO_AGGR_INVENTORY_DATA_FLG. The target tables to check are PMO_ACTIVITIES and PMO_CUM_SLS.
2. Create the forecast run corresponding to the forecast run type and change the settings as needed. The user can select **Submit**, which will start training the model.
3. Once the forecast run is complete, the parameters are generated. If the user is satisfied with the parameters, then the user must Approve Demand Parameters, Approve Base Demand and Forecast. Then the user must activate the forecast run type and map it to the application. (In this case, it is LPO.)

Optimization

Once the forecasting run is complete, the next step is to supply the data for optimization. This data must be supplied before the user can optimize the runs to generate recommendations. To load this data, the user must use the PRO_MASTER_ADHOC job with the appropriate flags as described here: <https://docs.oracle.com/en/industries/retail/retail-insights-cloud/21.0/rapog/mstr-dat-load-oo-dsgn-ovrvw.htm>

1. **Configurations.** The following configurations must be set up appropriately to supply the data at the relevant levels as well as to generate recommendations:
PRO_CUST_HIER_PROCESSING_LVL, PRO_OPT_LOC_REC_LVL, and
PRO_LOC_HIER_PROCESSING_LVL (which must be the same as
PRO_OPT_LOC_REC_LVL). When recommendations are to be set by price zones then this is set to the Company level. Values are PRO_OPT_MERCH_REC_LVL,

PRO_OPT_TIME_REC_LVL, PRO_PROD_HIER_PROCESSING_LVL (for example, class), PRO_PROD_HIER_RUN_SETUP_LVL (for example, dept), and PRO_PROD_HIER_TYPE (for example, 3 for extended hierarchy).

2. **Season-related data.** The target tables to check are PRO_SEASON, PRO_SEASON_PERIOD and PRO_SEASON_PRODUCT. It is not required to provide all historical seasons, as the optimization is concerned only with the ongoing seasons.
3. **Business Rules.** It is important that the rules are defined at the highest merchandise/location/season levels so that all items always find a rule to use. The target table to check is PRO_OPTIMIZATION_RULES.
4. **Price Ladder.** The target tables to check are PRO_PRICE_LADDER and PRO_PRICE_LADDER_DTL.
5. **Create LPO runs.** The user can run the PRO_OPT_CREATE_RUNS_ADHOC_JOB with the FORCE option to create the LPO runs for all merchandise and location/price zones.
6. **Optimize LPO runs.** The user can run the PRO_OPT_ADHOC_JOB with the -batch option to optimize all the above created runs.

Manage LPO Recommendations

The results will be available for the user to review/approve/export once the optimization job is complete.

Batch

Once the user is satisfied with the recommendations, then the user can put the system on a batch schedule to run at regular time intervals. Refer to "[POM Jobs](#)" about how to enable/disable the relevant jobs.

Security

LPOn supports both data-based and user role-based filtering or security. Data filtering informs the application regarding which merchandise and locations can be accessed by a particular user (for example, the user 'tom' can only access locations within US). The user role restricts or allows which UI actions can be performed by a particular user.

Data Filtering

Data filtering can be specified (and integrated wherever applicable) in a similar fashion as discussed in Chapter 4, "Manage Data Filtering" of *Oracle Retail Merchandising Suite - Administration Guide* (https://docs.oracle.com/cd/E79623_01/rms/pdf/192000/merchcs-admin.pdf). When a customer does not have Oracle Retail Merchandising Service, then the following interfaces can be leveraged to specify the user groups and users.

- RAF_FILTER_GROUP_MERCH
- RAF_FILTER_GROUP_ORG
- RAF_SEC_GROUP
- RAF_SEC_USER11. Pricing Groups. If the user to define pricing groups such as Same Markdown Discount, etc., then it must be provided here for batch LPO runs to pick up.¹² Flex Facts (or Custom Columns). If the user desires to see custom
- RAF_SEC_USER_GROUP

User Roles

User roles are used to set up application user accounts through Oracle Identity Management (OIM). See *Oracle Retail AI Foundation Cloud Services Administration Guide* for details. Five roles are supported in this application:

- Pricing Administrator
- Pricing Manager
- Pricing Analyst
- Buyer
- Targeted Offer Role

Data Input Requirements

This section provides information about setting up the data that the LPO application uses to generate optimal price recommendations, including guidelines on the expectations for the data element requested and where it is used. Information about these files can be found in *Oracle Retail Insights Cloud Service Suite/Oracle Retail Analytics and Planning Cloud Services Data Interface*.

Hierarchy Data

Hierarchies are part of the core data elements that are used in LPO, in both the forecasting and optimization modules. The four types of hierarchies are Location Hierarchy, Merchandise Hierarchy, Calendar Hierarchy, and Customer Segments Hierarchy. Hierarchy data is required.

Most of the interfaces have numerous columns and thus, it is possible to define which columns will be populated in the interface by specifying the column headers (order of columns does not have to be the same) in the corresponding file, such as PRODUCT.csv.ctx.

- Location Hierarchy. An example of location hierarchy is: CHAIN ' COUNTRY ' REGION ' DISTRICT ' STORE. When the user wants to optimize by location hierarchy, then the run's optimization level must match a node in the location hierarchy. For example, you can run the optimization at the District level. Note that the E-com channel can be defined as part of the location hierarchy. Relevant interfaces are W_INT_ORG_ATTR_D, W_INT_ORG_DH, and W_INT_ORG_D. These are all loaded using the ORGANIZATION.csv file.
- Price Zone Group and Price Zones. LPO provides the user with the flexibility to group any set of stores as part of a price-zone and use it as the run setup level for optimization. A price-zone group contains the set of price-zones, and each price-zone contains the set of stores. Note that two price-zones with a price-zone group cannot have any overlapping stores within a price-zone group. An example of Price-zone Group (PZG) and Price-zones (PZ) is as follows: PZG-100, mapped to DEPT1 with PZ-101 = Stores 1 to 10, PZ-102 = Stores 11 to 20. PZG-200, mapped to DEPT2 with PZ-201 = Stores 1 to 5. PZ-202 = Stores 6 to 10. PZ-203 = Stores 11 to 20. Note that in LPO, a price-zone cannot be defined above country-level since the currencies can be different. Further, the price-zone can be associated with a list of items. Different merchandise in a Department at a particular store can be part of different price zones. For example, women's university apparel in all university-based stores is grouped into one price zone and the women's non-university apparel belongs to another price-zone. If a price-zones are not defined or used, then the UI displays a default value of -1-WORLDWIDE.

Corresponding interfaces are W_RTL_CLSTR_GRP_DS, W_RTL_CLSTR_GRP_HDR_LC_DS, W_RTL_CLSTR_GRP_PRD_DS, W_RTL_CLSTR_HDR_DS. In C_ODI_PARAM, when the value for CLSTR_PROD_LEVEL is in (DEPT, CLS, SBC) then all MERCH_ID values map to the associated level in W_PROD_CAT_DH. On the other hand, in C_ODI_PARAM, when the value for CLSTR_PROD_LEVEL is in (ITEMLIST), then all MERCH_ID values map to an Item List ID in W_RTL_ITEM_GRP1_D. The latter allows the retailer to provide any custom item lists to be mapped to a price zone.

- **Merchandise Hierarchy.** An example of merchandise hierarchy is as follows: CHAIN 'COMPANY/BANNER 'DIVISION ' DEPARTMENT 'CLASS ' SUBCLASS ' STYLE ' COLOR ' SIZE (SKU). There can be up to nine levels in the merchandise hierarchy. When LPO is used for fashion apparel, it is expected that the you will provide Style and Color through the relevant interfaces. However, if the retailer deals with merchandise such as electronics, that does not necessarily have style or color, the application will work but some of the UI functionality will not be helpful. For example, Custom Rule has the merchandise selectors Class, Subclass, and Style. If no styles are loaded, then the last selector will not be helpful. Further, in such a situation, the recommendation levels will be at SKU level, as other levels may not be meaningful.

The STYLE, COLOR and SIZE SKUs are expected to be provided in the W_PRODUCT_D interface, while the levels between CHAIN and Subclass are provided via W_PROD_CAT_DH interface. In addition, there must be consistency among the levels provided when using an extended hierarchy. W_PRODUCT_ATTR_D, is used to indicate the relationship between Style, Style/Color, and Style/Color/Size for an extended hierarchy. All of these tables will be populated when providing a PRODUCT.csv file as a foundation file input. If the retailer wants to use an extended merchandise hierarchy, then the retailer must populate this interface.

Here are the specific fields in W_PRODUCT_ATTR_D table that are used:

- PRODUCT_ATTR13_NAME = PROD_NUM for the Style (for example, 0000190086820900)
- PRODUCT_ATTR14_NAME = PROD_NUM for the Style/Color (for example, 190086834203)
- PRODUCT_ATTR15_NAME = PROD_NUM for the Style/Color/Size (for example, 1975699). The value of PROD_NUMs is the same as the value in the W_PRODUCT_DS.PROD_NUM interface.

Here is what this looks like:

PROD_NUM	PRODUCT_ATTR13_NAME	PRODUCT_ATTR14_NAME	PRODUCT_ATTR15_NAME
STYLE_PROD_NUM	STYLE_PROD_NUM		
STY/COL_PROD_NUM	STYLE_PROD_NUM	STY/COL_PROD_NUM	
STY/COL/SIZ_PROD_NUM	STYLE_PROD_NUM	STY/COL_PROD_NUM	STY/COL/SIZ_PROD_NUM

- **Calendar Hierarchy.** This is one of the core hierarchies. The retailer can specify the calendar depending on their business requirements (for example, fiscal calendar). The relevant interface for this is W_MCAL_PERIOD_D, loaded using CALENDAR.csv file.
- **Customer Segments Hierarchy.** LPO supports pricing recommendations at the customer segment level. To use this functionality, the retailer must load customer segments. The relevant interfaces to supply this data are W_RTL_CUSTSEG_DS and W_RTL_CUST_CUSTSEG_DS. Retailers can use customer segments for many purposes;

however, for LPO purposes, retailers can use one set of customer segments to determine forecasting parameters and optimal price recommendations.

Runs in the LPO can be set up to run at configured levels for location or price zone, merchandise, calendar, and customer levels.

- **Location Level.** The run's location processing level is configurable in RSE_CONFIG, and it is denoted as PRO_LOC_HIER_PROCESSING_LVL. For example, a typical level is Region. If the user specifies price-zones, then location level defaults to COUNTRY.
- **Merchandise Level.** The run's merchandise processing level is configurable in RSE_CONFIG, and it is denoted as PRO_PROD_HIER_PROCESSING_LVL. For example, a typical level is the Class level.
- **Calendar Level.** The run's calendar processing level is configurable in RSE_CONFIG, and it is denoted as PRO_CAL_HIER_PROCESSING_LVL. For example, a typical level is Week.
- **Customer Level.** The run's customer processing level is configurable in RSE_CONFIG, and it is denoted as PRO_CUST_HIER_PROCESSING_LVL. For example, a typical level is Segment or Chain (that is, segment-all).
- **Setup Level.** The run is set up at a level higher than the run's merchandise level. This is configurable in RSE_CONFIG, and it is denoted as PRO_PROD_HIER_RUN_SETUP_LVL. For example, a typical level is Department level.

In this guide, the typical processing levels are generally used as illustration. If necessary, further details will be used to clarify non-typical levels. These levels are also important for the forecasting module since it provides the relevant parameters at the appropriate levels so that it can be consumed by the optimization module. It is essential that the configuration levels be decided based on the business of the retailer at the time of implementation.

Hierarchies are assumed to be set up once for each season and can be revisited whenever it changes. It is expected that the hierarchies do not change from one week to next week within a year (or season). Since hierarchies are core foundational elements any change to hierarchies can be costly and must be planned accordingly. For example, if customer segments are redone, then the parameters must be recalculated. The retailer must plan for such changes as such changes cannot necessarily be completed in the normal weekly batch process or window.

Inventory Data

Required interface. Historical inventory data as well as daily inventory data can be specified through this interface: INVENTORY.csv (staging table W_RTL_INV_IT_LC_DY_FS) . In addition to the non-null fields, the minimum required data fields that used by LPO are:

- INV_SOH_QTY - store on hand
- INV_ON_ORD_QTY - on order inventory quantity
- INV_IN_TRAN_QTY - in transit inventory quantity
- INV_UNIT_RTL_AMT_LCL

It is also recommend populating some other core fields on the table if the data exists, such as the RTL and COST values:

- INV_SOH_RTL_AMT_LCL - total retail value
- INV_ON_ORD_RTL_AMT_LCL - total on-order value
- INV_IN_TRAN_RTL_AMT_LCL - total in-transit value
- INV_AVG_COST_AMT_LCL - weighted avg cost (WAC)

- INV_UNIT_COST_AMT_LCL - current unit cost
- INV_SOH_COST_AMT_LCL - total OH cost value
- INV_ON_ORD_COST_AMT_LCL - total on order cost value
- INV_IN_TRAN_COST_AMT_LCL - total in transit cost value
- DOC_CURR_CODE - document currency of the record in the source system
- LOC_CURR_CODE - local currency for the AIF data warehouse system
- LOC_EXCHANGE_RATE - exchange rate (for BCF probably hard-coded to 1)

LPO also uses these two fields, which cannot be provided directly through the above interface. Instead, they are calculated based on inventory receipts data provided in RECEIPT.csv file (W_RTL_INVRC_IT_LC_DY_FS staging table).

- LAST_INVRC_DT
- FIRST_INVRC_DT

Once the inventory data is loaded, the user can configure which inventory components are to be used in LPO by setting the corresponding flags appropriately: RSE_INV_QTY_BOH_FLG, RSE_INV_QTY_ON_ORD_FLG, RSE_INV_QTY_IN_TRANSIT_FLG.

Price History Data

Required Interface. Historical price data as well daily price history can be loaded using PRICE.csv file (W_RTL_PRICE_IT_LC_DY_FS staging table). In addition to non-null fields, these are the core fields to populate:

- PRICE_CHANGE_TRAN_TYPE. The PRICE_CHANGE_TRAN_TYPE is based on the RMS usage of the value in PRICE_HIST. For example, 0 = initial price, 2 = cost change, 4 = cost reg price, 8 = clearance price, 9 = promo price.
- SELLING_UNIT_RTL_AMT_LCL
- BASE_COST_AMT_LCL
- DOC_CURR_CODE
- LOC_CURR_CODE
- LOC_EXCHANGE_RATE

Retail Code

Required Interface. Loaded using the CODES.csv file. LPO directly uses W_RTL_CODE_D values to drive the markdown results, for example, if LPO is generating a first markdown then the associated code/desc values for the first markdown that customer wants to see are obtained from this table. An example of this data is provided below.

Table 13-1 Sample Retail Code Data

CODE_ TYPE	CODE	CODE_ SEQ	CREAT_ ED_BY_ ID	CHAN_ GED_ B Y_ ID	CREAT_ ED_ ON_ DT	CHAN_ GED_ O N_ DT	DELET_ E_ FLG	DATAS_ OURC_ E_ NUM_ ID	ETL_P_ ROC_ WID	INTEG_ RATIO_ N_ ID
MKDN	1	1	-1	-1	11/30/2020 12:42:40 PM	11/30/2020 12:42:40 PM	N	1	1	MKDN~1
	2	2	-1	-1	11/30/2020 12:42:40 PM	11/30/2020 12:42:40 PM	N	1	1	MKDN~2
	3	3	-1	-1	11/30/2020 12:42:40 PM	11/30/2020 12:42:40 PM	N	1	1	MKDN~3
	4	4	-1	-1	11/30/2020 12:42:40 PM	11/30/2020 12:42:40 PM	N	1	1	MKDN~4

Retail Sales Data

Retail Sales data is a required interface and is provided through SALES.csv file (W_RTL_SLS_TRX_IT_LC_DY_FS staging table). LPO requires you to provide retailer's historical sales data at the desired hierarchy levels. If you want to use the targeted offers, then you must provide the customer-linked transactions sales data. The retailer does not have to provide the data aggregated to the segment-level but must provide the customer ID and customer segment mapping as part of customer segment interface.

Note that the sales returns are handled in the same interface. The returns data identifies the original location where the item was purchased and when it was purchased. This information is useful in returns analysis and in calculating the returns parameters. If the user wants to separate the returns from gross sales, then retailer is required to provide these two values separately.

When the system is in production, the latest incremental sales data is obtained as part of the batch process.

Promotions

A historical and future planned promotions-related interface is optional. LPO allows the user to provide any promotions using the following interface: RSE_PLAN_PROMOTION_STG. This interface provides the retailer with the ability to specify the start and end dates of a promotion event and which merchandise or locations are part of the promotion. This information can help LPO in two ways: identify when the promotions have occurred in the past and learn which future planned promotions are similar to the ones in the past. In this interface, the retailer must specify name, when (time), which items, and how deep (for example, for a Father's Day Sale, 20% off all merchandise in Mens division and across all stores).

Warehouse Inventory Allocation

The Warehouse Inventory Allocation interface is optional. The LPO user can provide the warehouse to store/online location mapping through this interface (RSE_INV_WHSE_LC_PR_ALLOC_STG). In addition, the user can provide the allocation percentage for each item (that is, how much of warehouse inventory is allocated to the particular store/online location). The user can configure whether to include all components of warehouse inventory in the calculation: warehouse on order, warehouse in transit, warehouse on hand using the following flags: RSE_WHSE_INV_QTY_ON_ORD_FLG, RSE_WHSE_INV_QTY_IN_TRANSIT_FLG, and RSE_WHSE_INV_QTY_BOH_FLG. If the user does not provide the allocation percentages, then LPO provides the user with ability to virtually allocate warehouse inventory using an in-built optimization algorithm. The optimization algorithm determines the allocation percentages based on the maximization of revenue across all warehouses. To turn on the warehouse virtual allocation algorithm, corresponding flag RSE_PRO_WHSE_ALLOC_FLG needs to be set to Y.

Competitor Prices

The Competitor Price interfaces, which are optional, are used to provide competitor prices. The two interfaces are W_RTL_COMP_PRICE_IT_LC_DY_FS and W_RTL_COMP_STORE_DS. The retailer can specify the price of an item at multiple competitors/locations and, depending on the configured option, either the minimum or the average of competitor prices is used for the optimization. Competitor prices are used to restrict the price recommendations to a certain percent range of the competitor price. For example, if the retailer wants to match the competitor's promotion of 20 percent off but with some tolerance, then this interface allows the retailer to specify the competitor price through the interface. Tolerance is specified as a global parameter in RSE_CONFIG. Note that in this release, the product can support competitor prices only through an interface.

Product Attributes

The product attributes that are useful in LPO forecasting and in particular for targeted offers can be defined through this interface. Raw product attributes (for example, fabric, material, and so on) can be provided through this interface. Make sure the attribute values are as clean as possible. For example, COKE vs. COK vs. COEK must be normalized to COKE so that LPO does not interpret them as different attributes.

The retailer must use ATTR.csv and PROD_ATTR.csv files, which populates the table W_RTL_ITEM_GRP1_D, since it can support any number of attributes, such as BRAND, and the batch load process picks up the attributes added there automatically. The product attributes that are useful in LPO forecasting and in particular for targeted offers can be defined through this interface. Raw product attributes (for example, fabric, material, and so on) can be provided through this interface. Make sure the attribute values are as clean as possible. For example, COKE vs. COK vs. COEK must be normalized to COKE so that LPO does not interpret them as different attributes.

If the retailer wishes to load STYLE or COLOR as product attributes, then the W_RTL_ITEM_GRP1_D interface is used to provide style and color attributes for the different products, using a value in PROD_GRP_TYPE of either STYLE or COLOR. The actual values for the Styles and Colors are provided in columns FLEX_ATTRIB_2_CHAR and FLEX_ATTRIB_4_CHAR using values that represent the ID for the Style (for example, 1234), and a description for the Style (for example, Loose Fit). The columns FLEX_ATTRIB_3_CHAR and FLEX_ATTRIB_10_CHAR contain the appropriate designation for STYLE or COLOR.

LPO supports grouping of product attribute values and it is required to load these additional interfaces: RSE_PROD_ATTR_GRP_VALUE_STG and RSE_PROD_ATTR_VALUE_XREF_STG. It is not required to group the attribute values. Consider the following examples:

- Consider Brand attribute with values Adidas, Champion, Nike and Under Armour. Since we are not grouping any attribute values for attribute Brand, each value by itself will be a group.
- Consider the Color attribute. There might be too many variants of the same color and thus it might be meaningful to group the color values. The first interface defines how many groups of colors would be there - for example, Trending, Casual, Professional. The second interface maps the actual attribute values to the groups - for example, Trending = {Flame Scarlet, Biscay Green}, Casual={Pink, Green}, Professional={Black, White, Grey}.
- These interfaces can also help with normalization of attribute values. Consider an example when there are a lot of misspellings (for example, COEK vs. COK vs. COEK). The retailer can use this interface to normalize the attribute values and create one group called COKE.

Product Images Data

Product images that are available on a customer-hosted web server can be viewed in LPO via the Results screen. The PRODUCT.csv or W_RTL_PRODUCT_IMAGE_DS.dat interfaces contain a column called PRODUCT_IMAGE_ADDR, that can contain the full URL to an image of the product. This URL must be in the following format:

`http[s]://servername[:port]/location/filename.extension`

For example:

`PRODUCT_IMAGE_NAME = imagename.png`

`PRODUCT_IMAGE_ADDR = http://hostname/url/imagename.png`

`PRODUCT_IMAGE_DESC= Short description of the image`

The LPO application running in the cloud does not directly access these images, so there is no need to expose these images outside of the customer's firewall. As long as the user of the LPO application has access to the URL while running the LPO application, then the user's web browser will be able to resolve the URL and retrieve the images for display when they choose this option. The images must be in a file format that the web browser can display. Since the images shown in the UI are small, these images do not need to be high quality images. The size of the image files will affect the time it takes to render them.

Season

PRO_SEASON_STG is used to define the name of the season with the start and end dates. For example, Halloween Season is used to identify merchandise that sells for Halloween. If the retailer does not have any seasons, then the retailer can define one long season.

Season Periods

PRO_SEASON_PERIOD_STG defines which periods belong to the season and lets users define whether a promotion or a markdown is allowed in a particular period. This interface can include the promotion and markdown calendar information of the retailer.

Season Products

PRO_SEASON_PRODUCT_STG defines which products are associated with a season. This interface indicates which products are to be included in the optimization for recommendations.

Price Ladder

PRO_PRICE_LADDER_STG can be used to supply price ladders. It supports three types of price ladders: Price Point (denoted by type A), Percentage Off Ticket Price (denoted by PT), and Percentage Off Full or Original Price (denoted by PO). The LADDER_COUNT can be used to define the ladder for the first and/or subsequent markdowns or promotions. The Ladder_count=0 must always be defined and is used in optimization. Any non-zero ladder_count ladders (only type A) are used for post-processing of price recommendations. For example, Ladder_count=0 is ending with .99, and ladder_count=2 is ending with .98 and ladder_count=3 is ending with .97.

Currency

PRO_COUNTRY_LOCALE_STG can be used to define the currency based on the country. This allows the same instance to support multiple currencies. For example, US locations must generate price recommendations (and price-related metrics) in USD, and Canadian locations must generate price recommendations (and price-related metrics) in CAD. Budgets can be specified in one global currency and optimization runs can be in a different local currency based on PRO_COUNTRY_LOCALE_STG. In which case, budgets are converted based on exchange rates provided in W_EXCH_RATE_GS.

Holidays

RSE_HOLIDAY_STG can be used to define the holidays that are used by the retailer. Major holidays can cause additional traffic visiting the store or the web channel.

Pricing Product Groups

W_RTL_ITEM_GRP1_D (loaded using PROD_ATTR.csv) can be used to define the different pricing product groups that are used by the retailer. PRO_OPTIMIZATION_RULES_STG allows the retailer to specify the group types for each of the defined groups. For example, both these interfaces help retailer define which items comprise the group and that this group would need the same markdown discount. Such groups and corresponding constraints are picked up by batch optimization runs.

Budget

W_RTL_PLAN1_PROD1_LC1_T1_FS can be used to specify the budgets for markdowns and/or promotions used by the retailer. A budget can be fed in at the run's merchandise setup-level (for example, Division) or it can be fed in at the run's merchandise processing-level (for example, Department). The interface allows the user to provide separate budgets or a combined budget for promotions and markdowns. The relevant fields from the interface are: slspr_rtl_amt (promotion budget), slscl_rtl_amt (clearance budget), loc_exchange_rate, mcal_wid, delete_flg, prod_dh_wid, org_dh_wid. On loc_exchange_rate, if the retailer operates in only one country, then it can be 1. However, it is possible that when a customer spans multiple countries, the planning budget can be in one global currency and when it is applied, it

can be in local currency. For example, to convert USD into GBP, set `loc_exchange_rate` as 1/0.75.

Strategy and Business Rules

`PRO_OPTIMIZATION_RULE_STG` allows the user to define business rules and associate them with a strategy set. This interface provides the following abilities for the user:

- There will always be a `DEFAULT_SET` strategy that is reserved to be used by batch runs in production. Other strategy sets are available for the user to do what-if runs.
- It is advised that retailers define all the business rules in the `DEFAULT_SET` strategy. It is not required that the user define complete set of rules again for a new strategy set. For example, an Aggressive Promotion Strategy set can define rules associated with promotions only. In such a case, the system will try to find the closest rule that is available in `DEFAULT_SET` when it is not available in the current strategy set.
- In any strategy set, the user can define rules at any merchandise and location levels. Rules will be applied to the merchandise and location that are closest to it. The term "Closest" can be defined as follows: for a given product level, move up the location hierarchy first to find any applicable rule. For example, the user defines Min First Markdown as 10% at Banner- All locations level and 20% for Men's Division-All locations. Then, merchandise in Men's division receives the 20% Min First Markdown rule, and all other merchandise receives the 10% Min First Markdown rule for all locations.
- Strategies can also be defined for price zone group (PZG) or price zone (PZ) level, when applicable. Note the following: (1) When the user wants to use PZG or PZ to specify rules, then the `LOCATION_KEY` must be -1 and `LOCATION_HIER_LEVEL` must be ALL. (2) When the user wants to specify a rule for all PZ Groups, then `PZ GROUP` = ALL (and `PZ Key`=-1). (3) When the user wants to specify a rule for all PZ within a PZG, then `PZ GROUP` must be specified (and `PZ Key`=-1). (4) When the user wants to specify a rule for a specific PZ, then `PZ GROUP KEY` = and `PZ KEY` must be specified. (5) When the user is not using PZG/PZ, then `PZ GROUP KEY` and `PZ KEY` must be -1.

Custom Columns

`W_RTL_FLEXFACT1_FS` can be used to add any custom column to the recommendations in the Manage Recommendation screen. Custom columns are picked up by batch runs at the time of their creation and assigned to the item and location. It supports up to ten numeric columns (for example, last year sales), twenty currency columns (for example, revenue till date), ten text columns (for example, supplier), five percentage columns (for example, sell-through for last four weeks), and five date columns (for example, last markdown date).

External Forecast Adjustment

`PRO_FCST_EXT_EFFECT_ADJ_STG` is used to provide adjustments to the forecast. It may occasionally be possible that extraneous factors that are not visible to LPO can impact the forecast (for example, Covid-19). In such a case, this interface can be leveraged to provide such factors to impact the LPO forecast.

Configuration and Expected Levels for Interfaces

The expected levels for the interfaces are based on the configuration levels specified for the recommendation and optimization levels. There are five dimensions that are used to define these configurations: Customer (applicable only for targeted offers), Calendar, Merchandise, Location, and Price-zone. The optimization level determines the levels at which optimization is

performed. For example, Division and Region (it can also be a price-zone). The recommendation level determines at what levels recommendations must be generated. Typically, these are lower levels of merchandise and location than the optimization levels (for example, Style/Color and Region). Note that the Calendar dimension tells the system at which level recommendations are generated (for example, Weekly vs Daily). The customer dimension tells the system that customer segments are available to generate targeted offers. Promotions and Markdowns are not generated at the customer segment level. If price-zones are not used, then the Price-zone column will be -1. When price-zones are used, then both optimization and recommendation levels will be price-zone. In such a case, both the location configurations, PRO_LOC_HIER_PROCESSING_LVL and PRO_OPT_LOC_REC_LVL, must be set to COUNTRY.

Table 13-2 Configuration: Expected Levels

Data	Configuration Parameters	Customer Segment	Calendar	Merchandise	Location	Price Zone
Optimization Level	Levels at which the optimization process is executed (Units of work)	PRO_CUST_HIER_PROCESSING_LVL	PRO_CAL_HIER_PROCESSING_LVL	PRO_PROD_HIER_PROCESSING_LVL	PRO_LOC_HIER_PROCESSING_LVL	-1 or PZ
Recommendation Level	Level at which the recommendation results are generated	PRO_OPT_CUST_REC_LVL	PRO_OPT_TIME_REC_LVL	PRO_OPT_MERCH_REC_LVL	PRO_OPT_LOC_REC_LVL	-1 or PZ

Depending on the above configuration levels, the interface levels expected are as follows:

Table 13-3 Interfaces: Expected Levels

Data	Interface Name	Notes	Customer	Calendar	Merchandise	Location	Price Zone
Season	PRO_SEASON_STG	Range of time periods (e.g., weeks) for the season. This has to match the calendar hierarchy loaded	N/A	N/A	N/A	N/A	N/A

Table 13-3 (Cont.) Interfaces: Expected Levels

Data	Interface Name	Notes	Customer	Calendar	Merchandise	Location	Price Zone
Season Period	PRO_SEASON_PERI OD_STG	Periods (e.g., weeks) within a season where the PRO_PROD_HIER_PROCESSING_LVL (e.g., CLASS) is active	N/A	PRO_CAL_HIER_PROCESSING_LVL Date range could span multiple calendar units (i.e. Multiple weeks)	PRO_PROD_HIER_PROCESSING_LVL or any hierarchy level higher	PRO_LOC_HIER_PROCESSING_LVL or any other higher hierarchy level	-1 or PZ
Season Product	PRO_SEASON_PRODUCT_STG	Products that must be included in given selling seasons	N/A	PRO_CAL_HIER_PROCESSING_LVL Date range can span multiple calendar units (i.e. Multiple weeks)	PRO_OPT_MERCH_RE_C_LVL	PRO_OPT_LOC_RE_C_LVL	-1 or PZ
Price Ladder	PRO_PRICE_LADDER_STG	Product price ladder	N/A	N/A	PRO_PROD_HIER_PROCESSING_LVL or any higher	PRO_LOC_HIER_PROCESSING_LVL	-1 or PZ
Country Locale	PRO_COUNTRY_LOCALE_STG	Defines the specific locale for the location. Provided at the COUNTRY level	N/A	N/A	N/A	PRO_LOC_HIER_PROCESSING_LVL or any hierarchy level higher	N/A
Planned Promotion	RSE_PLANN_PROMOTION_STG	Planned promotion at PRO_PROD_HIER_PROCESSING_LVL (e.g., CLASS)	N/A	PRO_CAL_HIER_PROCESSING_LVL Date range could span multiple calendar units (i.e. Multiple weeks)	PRO_PROD_HIER_PROCESSING_LVL	PRO_LOC_HIER_PROCESSING_LVL	-1 or PZ
Holidays	RSE_HOLIDAY_STG	Major holidays	N/A	Date when the holiday is observed	Any level in merchandise hierarchy	Any level in location hierarchy	N/A

Table 13-3 (Cont.) Interfaces: Expected Levels

Data	Interface Name	Notes	Customer	Calendar	Merchandise	Location	Price Zone
Strategy and Business Rules	PRO_OPTIMIZATION_RULE_STG	Business rules and associated strategies	N/A	N/A	Any level in merchandise hierarchy	Any level in location hierarchy	N/A

Table 13-4 Expected Levels for Optional* Interfaces

Data	Interface Name	Notes	Customer	Calendar	Merchandise	Location	Price Zone
Inventory	RSE_INV_PR_LC_WK_A_STG	Inventory available at the start of the period (e.g., week). Inventory is aggregated to the PRO_LOCATION_HIER_PROCESSING_LVL (e.g., Price Zone)	N/A	Week	SKU level (Style/Color/Size)	Store	N/A
Price Cost	RSE_PRICE_COST_PR_LC_WK_STG	Price cost for the time period (e.g., week)	N/A	Week	SKU level (Style/Color/Size)	Store	N/A

Optional* Interfaces

The interfaces specified in this section are not required and are available to handle special retailer situations. When the retailer provides the data through the interfaces tagged as 'required', then the system automatically uses the primary interfaces.

- Inventory (RSE_INV_PR_LC_WK_A_STG)
- Price Cost (RSE_PRICE_COST_PR_LC_WK_STG)

Integration with Retail Pricing Cloud Service

Additional configuration parameters must be set to integrate LPO with Retail Pricing Cloud Service. See the list of parameters shown in [Table 13-5](#). In addition, the Pricing CS username and password must be added under the credential store for Merchandise Pricing CS Access. You can modify these values using the UI screen for Control and Tactical Center -> Manage Credential Stores. Control and Tactical Center is available under the task menu.

Setting up the Configuration Parameters for Integration with Retail Pricing Cloud Service

Table 13-5 Configuration Parameters for Integration with Retail Pricing CS

Parameter Name	Parameter Value	Description
PRO_MERCH_RPM_FLG	N	Flag to identify whether to publish markdown recommendations to RPM.
PRO_MERCH_RPM_HOSTNAME		URL host name to publish recommendations to RPM. This must be same as CN name in the certificate.
PRO_MERCH_RPM_CONTEXT_ROOT	RpmRestService/services/private	Context root of the RPM webservice url.
PRO_MERCH_RPM_CLEARANCE_PATH	clearance/induction	Path of the webservice, e.g., clearance/induction.
PRO_MERCH_RPM_PORT	443	Port number to publish recommendations to RPM.
PRO_MERCH_RPM_HTTP_PROXY_HOSTNAME		Proxy host name to publish recommendations to RPM.
PRO_MERCH_RPM_HTTP_PROXY_PORT		HTTP proxy port number to publish recommendations to RPM.
PRO_MERCH_RPM_HTTPS_PROXY_PORT		HTTPS proxy port number to publish recommendations to RPM.
PRO_MERCH_RPM_VERSION	19.0N	RPM version to use for publishing customer segment to relate (format 16.0).
PRO_MERCH_RPM_USE_HTTPS	Y	Flag to identify whether to publish markdown recommendations to RPM over HTTPS.
PRO_MERCH_RPM_PROMOTION_PATH	promotion	Path of the RPCS webservice for promotions.

Figure 13-3 Setting up the Pricing CS Access in the Credential Store

ORACLE Science Cloud Services

Recommendations x Manage Credential Stores x

Select Credential Store

Select Credential Store: Merchandise RPM Access

Select Credential Store Map: oracle.retail.rse.merch.rpm.rest.service

Select Credential Store Key: oracle.retail.rse.merch.rpm.rest.service.credkey

Username *

Password *

Confirm Password *

Description

Create

POM Jobs

This section describes the batch jobs provided in the POM. Use this information to set up appropriate jobs for the regular batch process. Batch jobs do not expect any user input, and parameters are set to default values. Jobs are listed here by the functional area:

Foundation jobs that load product hierarchy, location hierarchy, calendar, product attributes, price zones, specify merchandise and location base user data filters (or security) are:

- ORASE_START_BATCH_JOB
- ORASE_START_BATCH_REFRESH_RESTR_JOB
- ORASE_START_BATCH_SET_ACTIVE_JOB
- ORASE_START_BATCH_END_JOB
- RSE_WEEKLY_INPUT_FILES_START_JOB
- WEEKLY_INPUT_FILES_WAIT_JOB
- WEEKLY_INPUT_FILES_VAL_JOB
- WEEKLY_INPUT_FILES_COPY_JOB
- RSE_WEEKLY_INPUT_FILES_END_JOB
- RSE_DIMENSION_LOAD_START_END_JOB
- RSE_DIMENSION_LOAD_START_START_JOB
- RSE_REGULAR_MAIN_LOAD_JOB
- RSE_LOC_SRC_XREF_LOAD_JOB
- RSE_PROD_SRC_XREF_LOAD_JOB
- RSE_LOC_HIER_LOAD_JOB
- RSE_LOC_HIER_TC_LOAD_JOB
- RSE_LOC_HIER_DH_LOAD_JOB
- RSE_FISCAL_MAIN_LOAD_JOB
- RSE_PROD_HIER_LOAD_JOB
- RSE_PROD_TC_LOAD_JOB
- RSE_PROD_DH_LOAD_JOB
- RSE_PROD_GROUP_LOAD_JOB
- RSE_PRICE_ZONE_LOAD_START_JOB
- RSE_PRICE_ZONE_GRP_LOAD_JOB
- RSE_PROD_ATTR_GRP_VALUE_STG_JOB
- RSE_PRICE_ZONE_LOAD_JOB
- RSE_PROD_ATTR_GRP_VALUE_LOAD_JOB
- RSE_PRICE_ZONE_LOAD_END_JOB
- RSE_PROD_ATTR_VALUE_XREF_STG_JOB
- RSE_DIMENSION_LOAD_END_START_JOB
- RSE_PROD_ATTR_VALUE_XREF_LOAD_JOB

- RSE_DIMENSION_LOAD_END_END_JOB
- RSE_SLS_START_START_JOB
- RSE_USER_DATA_FILTER_START_JOB
- RSE_SLS_START_END_JOB
- RSE_SLS_TXN_START_JOB
- RSE_USER_DATA_FILTER_LOAD_JOB
- RSE_LIKE_LOC_STG_JOB
- RSE_USER_DATA_FILTER_END_JOB
- RSE_LIKE_LOC_LOAD_JOB
- RSE_SLS_TXN_SETUP_JOB
- RSE_SLS_TXN_PROCESS_JOB
- RSE_SLS_TXN_END_JOB
- RSE_WKLY_DAILY_PROFILE_AGGR_PROCESS_JOB
- RSE_WKLY_DAILY_PROFILE_AGGR_START_JOB
- RSE_WKLY_SLS_CUST_SEG_START_
- RSE_WKLY_SLS_START_JOB
- RSE_WKLY_SLS_SEG_SETUP_JOB
- RSE_WKLY_SLS_SETUP_JOB
- RSE_WKLY_SLS_PROCESS_JOB
- RSE_WKLY_SLS_SEG_PROCESS_JOB
- RSE_WKLY_SLS_END_JOB
- RSE_WKLY_AGGR_SLS_START_JOB
- RSE_WKLY_AGGR_SLS_END_JOB
- RSE_WKLY_DAILY_PROFILE_AGGR_END_JOB
- RSE_WKLY_SLS_CUST_SEG_END_JOB
- RSE_SLS_END_START_JOB
- RSE_SLS_END_END_JOB
- RSE_INV_LOAD_START_JOB
- RSE_PRICE_LOAD_START_JOB
- RSE_INV_PR_LC_WK_STG_JOB
- RSE_PRICOST_PR_LC_WK_STG_JOB
- RSE_PRICOST_PR_LC_WK_LOAD_JOB
- RSE_PRICOST_PR_LC_WK_SETUP_JOB
- RSE_PRICOST_PR_LC_WK_PROCESS_JOB
- RSE_PRICE_LOAD_END_JOB
- RSE_INV_PR_LC_WK_LOAD_JOB
- RSE_INV_PR_LC_WK_SETUP_JOB
- RSE_INV_PR_LC_WK_PROCESS_JOB

- RSE_INV_LOAD_END_JOB
- ORASE_END_START_JOB
- ORASE_END_RUN_DATE_UPDT_JOB
- ORASE_END_END_JOB

Estimation and forecasting jobs that perform parameter estimation and generate forecast are:

- PRO_BASELINE_LOAD_START_JOB
- PRO_FCST_EXT_EFF_ADJ_START_JOB
- PRO_BASELINE_STG_JOB
- PRO_FCST_EXT_EFF_ADJ_STG_JOB
- PRO_PLAN_PROMOTION_LIFT_STG_JOB
- PRO_BASELINE_LOAD_JOB
- PRO_DOW_PROF_START_JOB
- PRO_FCST_EXT_EFF_ADJ_LOAD_JOB
- PRO_BASELINE_LOAD_END_JOB
- PRO_DOW_PROF_STG_JOB
- PRO_DOW_PROF_LOAD_JOB
- PRO_DOW_PROF_END_JOB
- PRO_FCST_EXT_EFF_ADJ_END_JOB
- PRO_LIFECYCLE_FATIGUE_LOAD_START_JOB
- PRO_MODEL_DATES_LOAD_START_JOB
- PRO_LIFECYCLE_FATIGUE_STG_JOB
- PRO_MODEL_DATES_STG_JOB
- PRO_LIFECYCLE_FATIGUE_LOAD_JOB
- PRO_MODEL_DATES_LOAD_JOB
- PRO_LIFECYCLE_FATIGUE_LOAD_END_JOB
- PRO_MODEL_DATES_LOAD_END_JOB
- PRO_PLAN_PROMOTION_LIFT_LOAD_JOB
- PRO_SEASONALITY_STG_JOB
- RSE_HOLIDAY_LOAD_START_JOB
- RSE_PLAN_PROMO_LOAD_START_JOB
- RSE_HOLIDAY_LOAD_STG_JOB
- RSE_HOLIDAY_LOAD_JOB
- RSE_HOLIDAY_LOAD_END_JOB
- RSE_PLAN_PROMO_STG_JOB
- RSE_PLAN_PROMO_LOAD_JOB
- RSE_PLAN_PROMO_LOAD_END_JOB
- PMO_HOLIDAY_LOAD_START_JOB

- PMO_HOLIDAY_LOAD_JOB
- PMO_HOLIDAY_LOAD_END_JOB
- PRO_SEASONALITY_LOAD_JOB
- PMO_ACTIVITY_STG_JOB
- PMO_ACTIVITY_LOAD_JOB
- PMO_RUN_EXEC_START_JOB
- PMO_CREATE_BATCH_RUN_JOB
- PMO_RUN_EXEC_SETUP_JOB
- PMO_RUN_EXEC_PROCESS_JOB
- PMO_CUMUL_SLS_START_JOB
- PMO_CUMUL_SLS_SETUP_JOB
- RSE_PUBLISH_FCST_PARAMETER_START_JOB
- PMO_CUMUL_SLS_PROCESS_JOB
- RSE_PUBLISH_FCST_PARAMETER_MODEL_DATE_JOB
- PMO_CUMUL_SLS_END_JOB
- RSE_MODEL_START_START_JOB
- RSE_MODEL_START_CALC_JOB
- RSE_PUBLISH_FCST_PARAMETER_SPREAD_DOWN_JOB
- RSE_PUBLISH_FCST_PARAMETER_SEAS_CURVE_JOB
- RSE_PUBLISH_FCST_PARAMETER_END_JOB
- RSE_MODEL_START_END_JOB
- PMO_RUN_EXEC_END_JOB
- RSE_FCST_BATCH_RUN_START_JOB
- RSE_CREATE_FCST_BATCH_RUN_JOB
- RSE_FCST_BATCH_PROCESS_JOB
- RSE_FCST_BATCH_RUN_END_JOB

Optimization jobs that load business rules and other related data are:

- PRO_COUNTRY_LOCALE_START_JOB
- PRO_OPT_RULE_START_JOB
- PRO_PLAN_PROMOTION_LOAD_START_JOB
- PRO_PROD_LOC_CDA_LOAD_START_JOB
- PRO_COUNTRY_LOCALE_STG_JOB
- PRO_OPT_RULE_STG_JOB
- PRO_COUNTRY_LOCALE_LOAD_JOB
- PRO_OPT_RULE_LOAD_JOB
- PRO_PROD_LOC_CDA_LOAD_JOB
- PRO_COUNTRY_LOCALE_END_JOB

- PRO_SEASON_LOAD_START_JOB
- PRO_OPT_RULE_END_JOB
- PRO_PLAN_PROMOTION_LOAD_JOB
- PRO_SEASON_STG_JOB
- PRO_PLAN_PROMOTION_LOAD_END_JOB
- PRO_SEASON_PRODUCT_STG_JOB
- PRO_SEASON_PERIOD_STG_JOB
- PRO_SEASON_CURR_OPT_METRIC_STG_JOB
- PRO_SEASON_PROD_MKDN_EDT_STG_JOB
- PRO_SEASON_LOAD_JOB
- PRO_SEASON_PRODUCT_LOAD_JOB
- RSE_BUDGET_START_JOB
- RSE_BUDGET_ALLOC_SETUP_JOB
- RSE_BUDGET_ALLOC_PROCESS_JOB
- RSE_BUDGET_END_JOB
- PRO_SEASON_PERIOD_LOAD_JOB
- PRO_SEASON_CURR_OPT_METRIC_LOAD_JOB
- PRO_SEASON_CURR_OPT_METRIC_SETUP_JOB
- PRO_SEASON_PROD_MKDN_EDT_LOAD_JOB
- PRO_SEASON_LOAD_END_JOB
- PRO_PROD_LOC_CDA_LOAD_END_JOB
- PRO_PRICE_LOAD_START_JOB
- RSE_WHSE_INV_ALLOC_LOAD_START_JOB
- PRO_PRICE_LOAD_SETUP_JOB
- RSE_WHSE_INV_ALLOC_STG_JOB
- PRO_PRICE_ELASTICITY_STG_JOB
- PRO_PRICE_LADDER_STG_JOB
- PRO_PRICE_LOAD_PROCESS_JOB
- PRO_PRICE_ELASTICITY_LOAD_JOB
- PRO_PRICE_LADDER_LOAD_JOB
- PRO_PRICE_LOAD_END_JOB
- RSE_WHSE_INV_ALLOC_LOAD_JOB
- RSE_WHSE_INV_ALLOC_PROCESS_JOB
- RSE_WHSE_INV_ALLOC_LOAD_END_JOB
- PRO_INVENTORY_LOAD_START_JOB
- PRO_INVENTORY_LOAD_SETUP_JOB
- PRO_INVENTORY_LOAD_PROCESS_JOB
- PRO_INVENTORY_LOAD_END_JOB

- PRO_OPT_START_JOB
- PRO_OPT_CREATE_RUNS_JOB
- PRO_OPT_JOB
- PRO_OPT_END_JOB
- PRO_SEASON_CURR_OPT_METRIC_PROCESS_JOB

Export and innovation workbench jobs that export the recommendations as well as grant permissions for objects in IW are:

- RSE_IW_GRANT_START_JOB
- RSE_IW_GRANT_JOB
- RSE_IW_GRANT_END_JOB
- RSE_EXPORT_START_START_JOB
- RSE_EXPORT_START_END_JOB
- PRO_EXPORT_START_JOB
- RSE_EXPORT_PREP_START_JOB
- PRO_EXPORT_PRICE_RECOM_JOB
- RSE_DAILY_BATCH_EXPORT_PREP_JOB
- PRO_EXPORT_RUN_RECOM_OPT_RESULT_JOB
- RSE_WEEKLY_BATCH_EXPORT_PREP_JOB
- PRO_EXPORT_RUN_TO_RESULT_JOB
- RSE_EXPORT_END_START_JOB
- RSE_QUARTERLY_BATCH_EXPORT_PREP_JOB
- PRO_TO_EXPORT_PRICE_RECOM_JOB
- RSE_EXPORT_END_END_JOB
- RSE_EXPORT_PREP_END_JOB
- PRO_EXPORT_END_JOB
- RSE_POST_EXPORT_DAILY_START_JOB
- RSE_POST_EXPORT_WEEKLY_START_JOB
- RSE_POST_EXPORT_DAILY_RESET_JOB
- RSE_POST_EXPORT_WEEKLY_RESET_JOB
- RSE_POST_EXPORT_DAILY_ZIP_JOB
- RSE_POST_EXPORT_WEEKLY_ZIP_JOB
- RSE_POST_EXPORT_WEEKLY_COPY_JOB
- RSE_POST_EXPORT_WEEKLY_FTP_JOB
- RSE_POST_EXPORT_WEEKLY_END_JOB
- RSE_POST_EXPORT_DAILY_COPY_JOB
- RSE_POST_EXPORT_DAILY_FTP_JOB
- RSE_POST_EXPORT_DAILY_END_JOB

Forecasting Science

LPO forecasting has two modules, parameter estimation and demand forecasting. Parameter estimation uses the historical data to determine the seasonality, markdown elasticity, promotion elasticity, and promotion lift (or holiday lift) values. Demand forecasting leverages the parameters estimated from historical data and in-season sales data (the base demand is re-estimated on a regular basis as part of batch process) to determine the demand and generate the forecast.

LPO forecasting supports the following features:

- Separate elasticity values for promotions and markdowns
- Customer segment level parameter values for seasonality, elasticity, and traffic lift for promotion and holidays
- The weather effects from historical data period can be incorporated into the estimation of parameters to remove the bias introduced by weather on forecasted sales.
- The ability to execute the following analysis:
 - Day of the week and time of the day profiles
 - Returns analysis

For both parameter estimation and demand forecasting stages, you can set up runs and configure various settings using the Manage Forecast Configurations functionality under Strategy & Policy Management. Based on the configuration settings, a batch job is used to initially estimate the parameters and then update them at specified intervals to reflect the latest sales trends. Similarly, a weekly batch job is used to generate demand forecasts every week after the weekly data has been updated.

Parameter Estimation

Parameter estimation consists of the following stages. Each stage has a series of configurable parameters and the corresponding default values. Modify the default values appropriately for each retailer. You can modify these settings using the Manage Forecast Configurations functionality under Strategy & Policy Management.

Overview of the Parameter Estimation Stages

The following sections describe the details of the settings in each stage.

1. **Data preparation.** Defines the levels to which data is aggregated on merchandise, location, customer segment, and time dimensions and the duration of the data used for parameter estimation.
2. **Preprocessing.** Filters the historical data and makes the first determination of item eligibility.
3. **Elasticity.** Determines the price elasticity for markdowns and promotions.
4. **Seasonality.** Estimates seasonality trends and identifies partitions with reliable seasonality.
5. **Promotion lift.** Estimates the traffic lift during promotion and holiday events. Corrects the seasonality curves to remove the traffic lift from promotion and holiday events.
6. **Output.** Generates parameter files in the format required by demand forecasting and offer optimization.

The parameter estimation stage generates markdown elasticity, promotion elasticity, seasonality and promotion lift values. Together they are referred to as demand parameters.

Parameter estimation requires at least 20 months of data. These requirements exist for the following reasons.

- The data from the beginning 4-6 weeks of the historical period must be removed from the data analysis so that the items that have truly started selling after the beginning of historical data period can be identified.
- The full life cycle data is required for all fashion/seasonal merchandise introduced during a fiscal year in order to estimate the seasonality curve for merchandise starting in every fiscal month. The least amount of data required is one year plus the typical season length of the merchandise.

Data Preparation

The data preparation stage defines the aggregation levels on merchandise, location, customer-segment, calendar dimensions, and duration of the data used for parameter estimation.

In addition to the above, configuration of the hierarchy level at which promotion data is processed is completed here.

Data Aggregation

Sales data, inventory data, and returns data are aggregated on the following dimensions: Merchandise, Location, Price Zone (if available), Customer Segment (if available), and Calendar. The hierarchy type ID corresponding to each dimension is configured using the following parameters in the RSE_CONFIG table. Use the Manage System Configurations functionality to set the appropriate ID for each dimension.

The level of aggregation for each dimension can be configured while setting up the forecast run types in Manage Forecast Configurations. In addition, while creating a run type, the user can indicate whether Price Zone and Customer Segment should be used as additional dimensions in the aggregation process.

- Merchandise (PMO_PROD_HIER_TYPE). Multiple hierarchy types are available for Merchandise, Product, and Extended Product hierarchies. Choose the ID corresponding to the appropriate product hierarchy. Typically, data is aggregated to Style/ Style-Color level on the merchandise hierarchy. Select the appropriate merchandise level when setting up the forecast run type in Manage Forecast Configurations.
- Location (PMO_LOC_HIER_TYPE). Multiple hierarchy types are available for location, for example, Location Hierarchy and Trade Area Hierarchy. Choose the ID corresponding to the appropriate location hierarchy. Typically, data is aggregated to Store Cluster/Price Zone level. Choose the appropriate level. Select the appropriate location level when setting up the forecast run type in Manage Forecast Configurations.
- Customer Segment (PMO_CUST_SEG_HIER_TYPE). Choose the hierarchy type ID associated with the customer segment. By default, customer segment data is aggregated at two levels, by individual customer segments and Segment-All, which includes all the segments. The user does not have the option to select the aggregation level by customer segment.
- Calendar (PMO_FISCAL_CAL_HIER_TYPE). Choose the hierarchy type ID associated with the Calendar hierarchy.

Table 13-6 Data Aggregation

Setting Name and Description	Default Value	Range of Values
Merchandise Hierarchy Type ID: ID corresponding to the merchandise hierarchy used for aggregation of sales, inventory, and returns data. PMO_PROD_HIER_TYPE	3	1 - 15
Location Hierarchy Type ID: ID corresponding to the location hierarchy used for aggregation of sales, inventory, and returns data. PMO_LOC_HIER_TYPE	2	1 - 15
Customer Segment Hierarchy Type ID: ID corresponding to the customer segment hierarchy used for aggregation of sales, inventory, and returns data. PMO_CUST_SEG_HIER_TYPE	4	1 - 15
Calendar Hierarchy Type ID: ID corresponding to the calendar used for aggregation of sales, inventory, and returns data. PMO_FISCAL_CAL_HIER_TYPE	11	1 - 15
Merchandise Level: Level in the merchandise hierarchy at which data is aggregated for sales, inventory, and returns. Typically, data is aggregated to STYLE-COLOR. Choose the appropriate level when setting up the forecast run type in Manage Forecast Configurations.	8	1 - 9
Location Level: Level in the location hierarchy at which data is aggregated for sales, inventory, and returns. Typically, data is aggregated to STORE CLUSTER. Choose the appropriate level when setting up the forecast run type in Manage Forecast Configurations.	5	1 - 6
Calendar Level: Level in the calendar hierarchy at which data is aggregated for sales, inventory, and returns. Choose the appropriate level when setting up the forecast run type in Manage Forecast Configurations.	4	1 - 5
Keep temporary tables: Turn this setting on to retain the intermediate tables generated during the parameter estimation process. PMO_KEEP_TEMP_TABLES	N	Y/N

Promotion Data Processing Level

Promotions data can be present at various levels in the merchandise and location hierarchy. Choose the level at which promotions are defined for estimating traffic lift values. For estimating traffic lifts, we consider major events which drive significant traffic across the chain. For example: there events are defined at country-banner level.

Table 13-7 Hierarchy Levels Selection

Setting Name and Description	Default Value	Range of Values
Promotion processing level Merchandise. Choose the ID corresponding to the merchandise level at which traffic driving promotions are defined. Usually this is Company/Banner if there are multiple banners within the company. PMO_MIN_LOC_HIER_PROCESSING_LVL	2	1 - 9
Promotion processing level - Location: Choose the ID corresponding to the location level at which traffic driving promotions are defined. Usually this is Chain/Country level. PMO_MIN_PROD_HIER_PROCESSING_LVL	2	1 - 6

Data Validation

Once the data preparation stage is complete, prepare data validation report and review it with the customer to confirm the data loaded matches with their expectations. Include the following in the data validation report.

- Summary of merchandise, location, customer segment, calendar hierarchies
- Sales units and amount by year at Chain and Division levels
- Summary metrics by Department-Price zone
- Sales, Inventory and Product count trends by Fiscal week and Fiscal year at Country-Banner level
- For the following data fields Ticket price, sales price, Item cost, Sales units, Sales amount, Inventory
 - Search for negative and null values
 - Significant percent of records with either negative or null values. Flag for review with customer.

Validating data with the customer is key to identifying potential data related problems early on. Investigate any potential data issues and work with the customer to resolve them. This ensures that the following stages are using the appropriate data.

Preprocessing

The Preprocessing stage filters the historical data to produce a subset of data that produces reliable demand parameters. The Preprocessing stage applies filters at the partition and week level. It performs the initial pruning of bad activity data. It does the first stage of determining eligibility and calculates certain values that can later be used in the calculation of elasticity and seasonality parameters. You can modify the configuration settings for the preprocessing stage using the Manage Forecast Configurations functionality under Strategy & Policy Management.

Data Filters Week Level

[Table 13-8](#) shows the various week level filters used in the Preprocessing stage. These filters are used to remove individual weekly records of data that do not meet the requirement defined as specified for each filter.

Table 13-8 Data Filters Week Level

Filter Name and Description	Default Value	Range of Values
Store count greater than 0. Used to filter activities with null or zero store count. IWF_STORE_GTR_ZERO	N	Y/N
SKU store count greater than 0. Used to filter activities with null or zero sku store count. IWF_SKU_STORE_GTR_ZERO	N	N/Y
Inventory data present. Used to indicate that the inventory data is available and reliable. If inventory data is available and reliable, then Preprocessing will use inventory data for filtering and other calculations. If inventory data is not available or unreliable, then it is ignored. IWF_INV_DATA_PRESENT	Y	N/Y
Sales unit threshold. Used to filter sales data records with sales units below threshold values. IWF_SALES_UNIT_THRESHOLD	1	Greater than or equal to 1.
Inventory threshold. Used to filter sales data records with inventory units below threshold values. IWF_INV_THRESHOLD	1	Greater than or equal to 0.
Use store count for Inventory threshold. Defines the criteria used for inventory threshold. Setting to Y will use Store count with inventory as inventory threshold criteria instead of actual inventory value. IWF_SKU_STORE_COUNT	N	Y/N
Threshold combination. Use AND/OR combination for Sales unit threshold and Inventory threshold. IWF_FILTER_COMBO	AND	AND/OR
Life cycle sell thru %. The item start and end dates are calculated using percentage sell through. The start date is the date when Life cycle sell through % (Start) is reached. The end date is the date when Life cycle sell through % (End) is reached. Life cycle sell through % (Start) is expressed relative to 0% so entering 2 means that the start date is when 2% of total sales has been achieved. Life cycle sell through % (End) is expressed relative to 100% so entering 2 means that the end date is when (100-2)% i.e., 98%, of total sales has been achieved. IWF_LOW_LCYCLE_SELL_PCT and IWF_HIGH_LCYCLE_SELL_PCT	Start: 2.0 End: 2.0	Start: greater than or equal to 0; less than or equal to 50. End: greater than or equal to 0; less than or equal to 50.
Relative price. Relative price thresholds are used to filter out item weeks with a ratio of sales price to maximum ticket price that fall outside the specified range for the start date and the end date. IWF_LOW_RELATIVE_PRICE and IWF_HIGH_RELATIVE_PRICE	Start: 0.2 End: 1.0	Start: greater than 0. End: greater than the start value.

Partition Filters

These filters are used to exclude all the weekly records from a Merchandise-Location-Customer segment partition at the aggregation level, for partitions with values that do not meet the requirement specified for each filter, as described below.

Table 13-9 Partition Filters

Filter Name and Description	Default Value	Range of Values
Minimum number of eligible weeks. Aa certain number of weeks are necessary in order to determine item eligibility. IF_MIN_ELIGIBLE_WEEKS	6	Greater than 0.
Minimum season (weeks). A certain season length (the number of weeks between the first and last activity) is required in order to determine item eligibility. IF_MIN_SEASON_WEEKS	6	Greater than 0.
Minimum sales units. The total number of units sold must be at least this value. IF_MIN_SALES_UNITS	10	Greater than 0.
Fraction of eligible weeks. The percentage of eligible weeks, expressed as a fraction of the season length. The season length is the number of weeks between the start and end dates. (See Life cycle sell thru % above.) IF_FRACTION_ELIGIBLE_WEEK	0.6	Greater than 0.0; less than or equal to 1.0.

Elasticity

The Elasticity stage determines the promotion and markdown elasticity values at the selected levels in the merchandise and location hierarchy. For a given merchandise and location hierarchy level, this stage also determines the elasticity value by customer segment. You can modify the configuration settings for the elasticity stage using the Manage Forecast Configurations functionality under Strategy & Policy Management.

In this stage, parameters must be set for

- Additional data filtering to reduce the noise in the elasticity estimation
- Merchandise and location hierarchy levels for which parameters are estimated
- Identifying markdowns weeks
- Identifying promotion weeks
- Reliability thresholds for elasticity estimation
- Transformation of elasticity values

Data Filters Weekly

Additional data filters are applied during the estimation of elasticity. These filters are used to reduce the noise from very deep price cuts or price cuts towards the end of life. An option is also available to the limit the data used for elasticity estimation.

Table 13-10 Data Filters Weekly

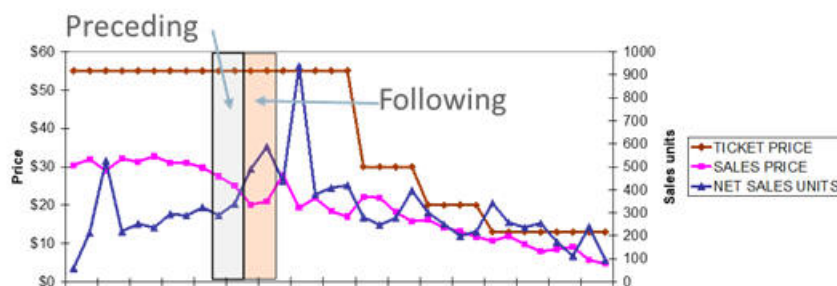
Filter Name and Description	Default Value	Range of Values
Relative price. Used to filter out item-weeks with a ratio of sales price to maximum ticket price that falls outside the specified range. WDF_RELPRICE_LOW and WDF_RELPRICE_HIGH	Low: 0.2 High: 1.0	Low ≥ 0 ; High $>$ low value
Relative inventory. The upper and lower bounds for the value for inventory relative to maximum inventory. WDF_PCTSALES_LOW and WDF_PCTSALES_HIGH	Low: 0.2 High: 1.0	Greater than or equal to 0. Low ≥ 0 High $>$ low value
Range filter. Used to eliminate unreliable data using start date and end date for the period. Both the start date and the end date are Null by default. A Null value means that the field is not used in the filter. One or both of the fields can have a value of Null. If both of the fields are Null, then the data is not filtered. WDF_RANGEFILTER_MINCALWK and WDF_RANGEFILTER_MAXCALWK	Start date: Null End date: Null	-

Hierarchy Levels Selection

Select the merchandise and location levels for which parameter estimation should calculate demand parameters. Parameter estimation calculates demand parameters for the partitions of the levels that are the cross-products of all the levels you select. The highest level is set based on PMO_MIN_LOC_HIER_PROCESSING_LVL (typically, Banner) and PMO_MIN_PROD_HIER_PROCESSING_LVL (typically, Department). The lowest level is set based on the aggregation levels that the user selects while creating the forecast run types. In Manage Forecast Configurations, the user can select from the list of levels that are between the highest and lowest levels.

Markdowns

The parameters described below are used to identify markdowns in the aggregated weekly sales data. During a markdown window, the application is interested in a drop in prices from the preceding window to the following window. During a promotion, the application is looking for similar prices during the preceding and following window but a drop in prices during the promotion window. Similar parameters are used to identify Promotions as well.

Figure 13-4 Markdown Window

Time window defines the weeks before and after the markdown. The preceding weeks value is the number of weeks before the markdown occurred. The following weeks value is the number of weeks after the markdown and includes the week of the markdown. A week is a calendar week. A time window that satisfies all the markdown parameter settings is classified as a markdown window.

Minimum eligible weeks is the minimum of the actual number of weeks in the Time Window that have data. The Time Window is in calendar weeks, and not every calendar week actually has sales data. So the actual number of weeks with data that are within the Time Window can actually be smaller than the Time Window.

Maximum deviation is the deviation of the sales price in the weeks before and after the markdown. Provides stability on the variance.

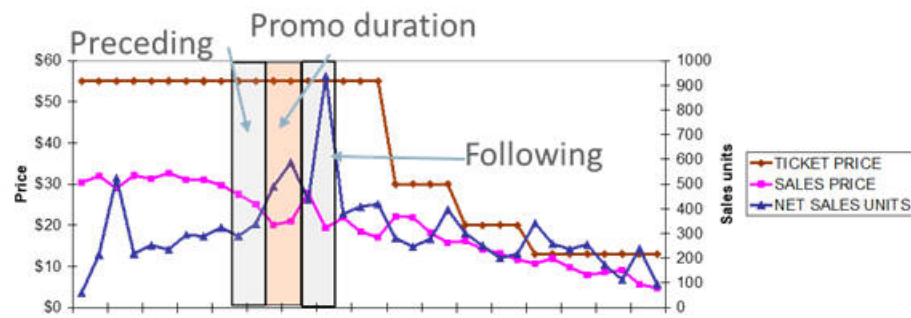
Min Markdown depth is the drop in average sales price from preceding weeks to the following weeks is higher than this threshold value. Markdown depth = 1- (Avg Price following weeks/Avg Price preceding weeks). In other words, this parameter controls how much of a price decrease is required in order for the price decrease to count as a markdown.

Table 13-11 Markdown Parameters

Name and Description	Default Value	Range of Values
Time Window	Preceding: 2	1–3
MKDN_TIMEWINDOW_PRECEEDING and MKDN_TIMEWINDOW_FOLLOWING	Following: 2	1–3
Eligible Weeks	Preceding: 2	1–3
MKDN_ELIGWKS_PREC and MKDN_ELIGWKS_FOLL	Following: 2	1–3
Maximum Deviation	Preceding: 0.1	0–1
MKDN_MAXDVT_PRECEEDING and MKDN_MAXDVT_FOLLOWING	Following: 0.1	0–1
Min Markdown Depth	0.1	0–1
MKDN_MINMKDN_DEPTH		
Use Ticket Price: Both Ticket price and sales price values from the preceding and following weeks satisfy min markdown depth criteria.	N	Y/N
USE_TICKET_PRICE		

Promotion One Week

For identifying promotions, a few additional parameters are defined below in addition to Time window, Eligible weeks, and maximum deviation, which are defined above.

Figure 13-5 Promotion Window

Min Promotion depth is the drop in average sales price during the week of promotion compared to preceding/following weeks. In other words, this parameter controls how much of a price decrease is required in order for the price decrease to count as a promotion. Time window defines the weeks before promotion, during promotions, and after the promotion. The preceding weeks value is the number of weeks before the markdown occurred. The following weeks value is the number of weeks after the markdown and includes the week of the markdown. A week is a calendar week. Time window that satisfies all the promotion parameter settings is classified as a promotion window. Promotion depth = $1 - (\text{Avg. Price during promotion weeks} / \text{Avg. price during preceding and following weeks})$. Time Window - Promo duration: Duration of the promotion event, this is set to 1 for 1-week promotion. Max deviation - Preceding/Following: Avg. price deviation across preceding and following weeks must be less than this threshold. This ensures that the sales price is similar in the weeks before and after the promotion.

Table 13-12 Promotion Parameters

Name and Description	Default Value	Range of Values
Time Window	Preceding: 1	1–3
PRWK1_TIMEWINDOW_PRECEEDING, PRWK1_TIMEWINDOW_FOLLOWING and PRWK1_TIMEWINDOW_PROMOWEEKS	Following: 1 Promo Duration: 1	1–3 1
Eligible Weeks	Preceding: 1	1–3
PRWK1_ELIGWKS_PREC and PRWK1_ELIGWKS_FOLL	Following: 1	1–3
Maximum Deviation	Preceding: 0.1	0–1
PRWK1_MAXDVT_PRECEEDING, PRWK1_MAXDVT_FOLLOWING and PRWK1_MAXDVT_PRECFOLLOW	Following: 0.1 Preceding/Following: 0.1	0–1 0–1
Min Promotion Depth	0.1	0–1
PRWK1_MINPROMO_DEPTH		

Promotion Two Week

Promotion duration parameter is set to 2 for identifying two week promotions. Promotion parameters for one-week and two-week promotions can be set to different values.

Table 13-13 Promotion Two Weeks

Name and Description	Default Value	Range of Values
Time Window	Preceding: 1	1–3
PRWK2_TIMEWINDOW_PRECEEDING, PRWK2_TIMEWINDOW_FOLLOWING and PRWK2_TIMEWINDOW_PROMOWEEKS	Following: 1 Promo Duration: 2	1–3 2
Eligible Weeks	Preceding: 1	1–3
PRWK2_ELIGWKS_PREC and PRWK2_ELIGWKS_FOLL	Following: 1	1–3
Maximum Deviation	Preceding: 0.1	0–1
PRWK2_MAXDVT_PRECEEDING, PRWK2_MAXDVT_FOLLOWING and PRWK2_MAXDVT_PRECFOLLOW	Following: 0.1 Preceding/Following: 0.1	0–1 0–1
Min Promotion Depth PRWK2_MINPROMO_DEPTH	0.1	0–1

Reliability Settings

Determine the partitions with reliable elasticity values.

Table 13-14 Reliability Settings

Name and Description	Default Value	Range of Values
Outlier threshold. Percentile threshold for removing outlier data points from elasticity estimation. Eg., Threshold is set to X%, Top X Percentile and Bottom X percentile sales ratio and price ratio data points are excluded from elasticity estimation. RS_OUTLIER_THRESHOLD	0.05	0–0.05
Eligible items threshold - Elasticity RS_ELIGIBLEITEMS_THRESHOLD	25	Greater than 1.
Std. Error threshold RS_STDERROR_THRESHOLD	0.3	0–1

Transformation

Transform the elasticity values obtained after the pruning based on reliability to cap the very low or very high elasticity values. The transformation settings also enable the user to shift the elasticity values to a desired range.

Table 13-15 Transformation

Name and Description	Default Value	Range of Values
Transform Percentile - Low TRANSPERCENTILE_LOW	0.1	0–1
Transform Percentile - High TRANSPERCENTILE_HIGH	0.9	0–1

Table 13-15 (Cont.) Transformation

Name and Description	Default Value	Range of Values
Elasticity range - Min: Select a value closer to the transform percentile - Low and higher than 1.2 ELASTICITYRANGE_MIN	1.3	Greater than 1.
Elasticity range - Max: Select a value closer to the transform percentile - high. ELASTICITYRANGE_MAX	2.8	Greater than elasticity range-min.

Seasonality

The Seasonality stage determines the seasonality values at the selected levels in the merchandise and location hierarchy. For a given merchandise and location hierarchy level, this stage also determines the seasonality value by customer segment. You can modify the configuration settings for the seasonality stage using the Manage Forecast Configuration functionality under Strategy & Policy Management.

In this stage, parameters must be set for

- Seasonality curve set up
- Reliability filters

Seasonality Curve Setup.

The Seasonality Curve Setup stage determines the season codes used for building seasonality curves, length of the seasonality curve, item count threshold, padding curve weight, and seasonality coverage of the final curves.

Season codes create additional partitioning in the dataset by introducing a time dimension. This stage can be used to partition seasonality curves by fiscal start month, fiscal start week, for example, Partition the Class-Store cluster-Customer segment data based on the merchandise time of introduction. Group merchandise based on the fiscal month of the start date. This ensures items starting in the same month receive the same season code. Weekly curves are useful for modeling short life cycle merchandise, while monthly curves are useful for medium to long life cycle curves.

Length of the seasonality curve: Seasonality curves are generated for 52 weeks. They can also be generated for longer duration if the merchandise life cycle length is longer. This also increases the duration of historical data required to prepare the seasonality curves.

Actual sales for a given merchandise-location-customer segment-Start month can be present for less than 52 weeks. To generate a curve for the entire 52 weeks duration, a padding curve is used.

Padding curve: The padding seasonality curve is determined by creating a basic curve for the highest merchandise/location partition. The final seasonality curve is calculated as $\text{weight} * \text{padding curve} + (1 - \text{weight}) * \text{seasonality curve}$.

Table 13-16 Seasonality Curve Setup

Name and Description	Default Value	Range of Values
Curve Type. Determines the duration of time partition based on fiscal week or fiscal month. CURVE_TYPE	Monthly	Monthly or Weekly
Basic Curve. Basic curves are generated by ignoring the partitioning on time dimension for a given merchandise-location-customer segment partition. BASIC_CURVE	Y	True or FalseY/N
Seasonality curve length. Determines the length of the final seasonality curves for regular (non-basic) season codes. The length from the start determines how many weeks after the start date are in the curve. SEASON_CURVE_LENGTH	52	Greater than or equal to 6.
Eligible items threshold. The minimum number of items that a Merchandise Hierarchy/Location Hierarchy/Customer Segment/Season Code partition must contain so that Raw-AP can produce a seasonality curve for the partition. ELIGIBLE_ITEM_THRESHOLD	5	Greater than or equal to 1.
Padding curve weight. The value used in determining the final seasonality curve. PADDING_CURVE_WEIGHT	0.4	Greater than 0.0; less than 1.0.
Basic curve start week. First week of the basic curve. This is represented as number of weeks from the most recent week with data. Should be greater than 52 and less than number of weeks of historical data.	57	52 - 80
Seasonality Coverage. First Fiscal year SEASON_COVERAGE_FIRST_FY	Latest fiscal year with historical data	Greater than or equal to 2020
Seasonality coverage. Last fiscal year SEASON_COVERAGE_LAST_FY	Latest fiscal year with historical data + 2.	Greater than or equal to 2020.

Reliability Filters

The filters in this stage are used to identify reliable seasonality curves based on the number of items in the partition, average sales volume, and season length threshold.

Table 13-17 Reliability Filters

Name and Description	Default Value	Range of Values
Keep top level curves. All the highest level curves are kept, regardless of threshold values. KEEP_TOP_LEVEL_CURVES	Y	Y/N
Prune curves with missing dates. Pruning of basic curves with missing dates is permitted. PRUNE_CURVES_MISSING_DATES	Y	Y/N

Table 13-17 (Cont.) Reliability Filters

Name and Description	Default Value	Range of Values
Eligible items threshold. Partitions with fewer numbers of eligible items than the threshold value are removed. Eligibility is defined during preprocessing.	5	Greater than or equal to 0.0.
Average weekly sales threshold. Partitions with average weekly sales below the threshold are removed. Weekly sales are the sum of all sales for all activities for a given week. AVG_WK_SALES_THRESHOLD	25	Greater than or equal to 0.
Raw seasonality length threshold. Curves can be discarded when the number of weeks from the first non-zero seasonality value to the last non-zero seasonality value is less than the threshold. In the seasonality stage, it is possible that many seasonality values of zero have been added to the curves. Note that basic season codes have a length of 53, so picking a value greater than 53 will prune out any basic season codes. Seasonality curves before padding are referred to as raw seasonality curves. RAW_SEASONALITY_LEN_THRESHOLD	10	Greater than or equal to 0.

Promotion

The promotion stage determines the traffic lift value associated with promotions and holiday events at the selected levels in the merchandise and location hierarchy. For a given merchandise and location hierarchy level, this stage also determines the promotion lift value by customer segment. A list of holidays and promotions, along with event start date and end date, is sent as a data feed.

In this stage, parameters must be set for determining the baseline to be used for estimation of lift values, Outlier threshold, min lift value, and max lift value. You can modify the configuration settings for the promotion stage using the Manage Forecast Configurations functionality under Strategy & Policy Management.

Table 13-18 Promotion

Name and Description	Default Value	Range of Values
Baseline. The baseline type determines how to smooth the seasonality curve. If you select the linear option, parameter estimation looks at event effective start date and event effective end date and draws a straight line between them. Effective start date is the weekend prior to the event start date and effective end date is the second weekend after the event end date. PROM_LIFT_BASELINE_TYPE	Linear	Linear, Min, or Max.
Outlier percentile. Very high lift values are flagged. PROM_LIFT_OUTLIER_PERCENTILE	0.05	0–1

Table 13-18 (Cont.) Promotion

Name and Description	Default Value	Range of Values
Lift value -High. Lift values flagged as outliers are capped to outlier percentile lift value. PROM_LIFT_HIGH	Y	Y/N
Lift value-min. Sets the threshold for min lift value. Any promotions with lift value lower than this threshold are not assigned any lift value. PROM_LIFT_MIN_VALUE	1.05	Greater than or equal to 1.

Output and Review of Parameters

The Propagation and Parameter Export to Target stages do not have separate settings. The Propagation stage populates the seasonality curves for all the years from the Seasonality coverage first fiscal year to the Seasonality coverage last fiscal year.

The Parameter Export to Target stage populates the parameters at the lowest level of estimation selected in the appropriate output tables. LPO typically expects the parameters at the lowest level of estimation. For example, LPO expects the demand parameters at Class-Store cluster level. Use escalation to fill in the parameters missing at Class-Store level. The Output stage escalates on location hierarchy first and then on merchandise hierarchy.

Run the parameter estimation by starting the batch process during the implementation phase. Details on the batch processing are covered in "[Batch Processing](#)". Perform the following checks after each stage in the parameter estimation is complete.

Elasticity (pmo_elasticity_parameters)

- Histogram of elasticity values, promotion vs. markdown: Distribution of elasticity values follows the expected trend. Typically follow a normal distribution around the median elasticity value.
- Elasticity escalation: Percentage of partitions receiving elasticity values from higher level.

Seasonality (pmo_seasonality_parameters; pmo_seasonality_curve_tbl)

- Review number of partitions impacted by reliability filters:
- Visually check the top-level seasonality curves
- Seasonality escalation: Percentage of partitions receiving elasticity values from higher levels.

Promotion lift (pmo_promtion_lift)

- Plot histogram of promotion lift values by event and compare across segments.

Day Level and Returns Metrics

This section covers the settings used for determining day of the week profiles, time of the day profiles, return rate, and average time to return.

Day of the Week Profiles

Day of the week profiles determine the relative sales strength across various days in a given week. These profiles are used to spread the forecasted demand to the day level. In order to

capture the variation across merchandise, location, segment, and calendar dimension, the configuration on each of the four dimensions is supported. In the current version, you can choose two configurable levels for estimating day of the week profiles. Level 1 corresponds to the estimation level at which profiles are calculated, and Level 2 profiles correspond to the escalated profiles. They are only used when profiles are not available at the estimation level.

Table 13-19 Day of the Week Profiles

Name and Description	Default Value	Range of Values
Merchandise level. ID corresponding to the merchandise level at which day of the week profiles are calculated. Usually at DEPARTMENT or higher. RSE_PMO_DLYPROF_MH_LVL_SQC1	4	1 - 9
Location level. ID corresponding to the location level at which day of the week profiles are calculated. Usually STORE CLUSTER or higher. RSE_PMO_DLYPROF_LH_LVL_SQC1	4	1 - 6
Merchandise level. Merchandise level - Escalated: ID corresponding to the higher merchandise level at which day of the week profiles are calculated. Usually at Division or higher. RSE_PMO_DLYPROF_MH_LVL_SQC2	3	1 - 3
Location level. Location level - Escalated: ID corresponding to the higher location level at which day of the week profiles are calculated. Usually Region or higher. RSE_PMO_DLYPROF_LH_LVL_SQC2	3	1 - 3
Min week. Calendar hierarchy ID corresponding to the oldest week used for day of the week profile calculation. PMO_DLYPROF_MIN_WK	User determined	
Max Week. Calendar hierarchy id corresponding to the latest week used for day of the week profile calculation. PMO_DLYPROF_MAX_WK	User determined	
Week count threshold. Partitions with data for less than the week count threshold are excluded from day of the week profile calculation. PMO_DLYPROF_WK_CNT_THLD	2	> 1
Sales threshold. Total sales from the merchandise, location, segment and calendar partition must be higher than this threshold value for the partition to be eligible for estimation of day of the week profile. PMO_DLYPROF_TOT_SLS_THLD	1000	Greater than 100.
Output level - Merchandise. Day of the week profiles are generated at this merchandise level for export. Any missing levels are filled using the higher level profiles.	4	1 - 9
Output level - Location. Merchandise: Day of the week profiles are generated at this location level for export. Any missing levels are filled using the higher level profiles.	4	1 - 6

Returns Metrics

For returns parameter estimation, the data is divided into two parts, one for building the model and the other for testing the model. Usually, one full year is used to build the model, and at least 13 weeks are used for testing the model. The most recent data corresponding to the build and test durations is used for returns modeling from the available data. Returns are modeled at various locations in the merchandise hierarchy, typically, Class, Department, and Division. For each of the merchandise levels, modeling is done for both customer segment and segment all levels on the customer segment dimension and at the data aggregation level on the location dimension. After applying the reliability filters, the missing parameters at the lowest merchandise level - location: data aggregation level - customer segment level are filled by escalation.

The return rate is estimated at the data aggregation level for the Merchandise-Location-Customer Segment level every week. This information, along with the returns parameters and sales forecast, is used to generate the returns forecast.

Table 13-20 Returns Metrics

Name and Description	Default Value	Range of Values
Merchandise level 1. The topmost merchandise level used for returns parameters. Typically Division or Banner. Choose the merchandise hierarchy ID corresponding to this level. PMO_RETURN_PART_MH_LVL1	2	1 - 6
Merchandise level 2. The intermediate merchandise level used for returns parameters. Typically Department. Choose the merchandise hierarchy ID corresponding to this level. PMO_RETURN_PART_MH_LVL2	4	1 - 6
Customer segment level 1 Merchandise level 2. The lowest merchandise level used for returns parameters. Typically Class. Choose the merchandise hierarchy ID corresponding to this level. PMO_RETURN_PART_MH_LVL3	5	1 - 6
Duration of data for Build. Number of weeks of data to be used for building the model. PMO_RETURN_BUILD_MODEL_WKS	52	52 - 80
Duration of data for Test: Number of weeks of data to be used for testing the model. Most recent weeks are used for testing the model. PMO_RETURN_TEST_MODEL_WKS	13	13 - 26
Sales threshold. Merchandise-Location-Customer segment partitions with total sales lower than the threshold are excluded from returns parameter estimation. PMO_RETURN_LOW_SALES_THRESHOLD	10	> 10
Model fit threshold. Merchandise-Location-Customer segment partitions with adjusted r square value lower than the threshold are pruned. This eliminates partitions where the model fit is poor. PMO_RETURN_RELIABLE_ADJST_R_SQR_THS LD	0.5	> 0.3

Table 13-20 (Cont.) Returns Metrics

Name and Description	Default Value	Range of Values
Data sufficiency threshold. Row count: Merchandise-Location-Customer segment partitions: Total with data less than the row count threshold are excluded from returns parameter estimation. PMO_RETURN_RELIABLE_ROW_COUNT_THSL D	500	>100

Demand Forecasting

The following multiplicative demand model is used as follows:

$\text{Demand} = \text{Base demand} * \text{Seasonality} * \text{Price effect} * \text{Promotion Traffic Lift} * \text{Store count} * \text{Other Effects}$

Base demand is updated in season every week at the Optimization recommendation level, for example, Style Color-Store cluster-Customer segment level. Demand forecasting leverages the parameters estimated from historical data and in-season sales data to generate a demand forecast. Elasticity for determining price effect is assigned using merchandise and location hierarchy. For assigning seasonality curve and promotion lift, the start date is required to determine the right partition on time dimension in addition to merchandise and location hierarchy. Demand forecasting set up involves configuration for start date and demand strategy.

Demand forecasting stage leverages the parameters estimated from historical data and in-season sales data to generate a demand forecast. Demand forecast values along with historical parameters are sent to LPO for generating optimal price recommendations and targeted offers.

In this stage, the following parameters must be set by the user to determine the model start date, season code and demand model settings. For all the parameters prefixed with RSE, you can modify the values in the RSE_CONFIG table using Manage System Configurations. For other parameters, use the Manage Forecast Configurations functionality under Strategy & Policy Management to modify parameters for Base Demand estimation.

Note that the merchandise/location/calendar hierarchy types do not need to be set for base demand separately, as they will be the same as the values set for PMO_PROD_HIER_TYPE, PMO_LOC_HIER_TYPE, and PMO_FISCAL_CAL_HIER_TYPE. In addition, the merchandise/location/calendar levels for base demand estimation do not need to be set separately, as they will be the same as the aggregation levels that are selected when creating a run type (AGGR_PROD_HIER_LEVEL, AGGR_LOC_HIER_LEVEL, and AGGR_CAL_HIER_LEVEL).

Table 13-21 Demand Forecasting

Name and Description	Default Value	Range of Values
Merchandise level -Store Weights. The merchandise level at which store weights are calculated. Choose the ID corresponding to the merchandise level, typically at Sub Class.RSE_BD_SW_PROD_HIER_LEVEL	6	1 - 9

Table 13-21 (Cont.) Demand Forecasting

Name and Description	Default Value	Range of Values
Start date - sell thru %. The earliest week ending date when a Style-Color - Store Cluster combination (data aggregation level used for generating forecast and price recommendations) achieves 2% sell thru is used to determine its start date. The start date is used to determine the seasonality curve assigned to a style color - store cluster combination.	2%	0%–5%
OPTPARAM_PCT_SLS_THRESHOLD		
Start date - max weeks from first sale. After the first sale date if the 2% sell thru is not achieved before this threshold then first sale date + max weeks from first sale is used to define the start date.	3	1–5
OPTPARAM_WEEK_LIMIT		
Seasonality - Max weeks from start. This setting is used to switch from a fashion curve to a basic curve if an item lives longer than expected.	42	Greater than 35.
OPTPARAM_SEAS_CURVE_THRESHOLD		
Demand Interval. Number of in-season weeks to be used for demand forecast-ing.0 implies the use of entire history from current season to generate the demand forecast.	4	0–6
BD_PERIOD_LEN		
Demand Interval - Poisson. Number of in-season weeks to be used for demand forecasting using Poisson demand strategy.0 implies use entire history from current season to generate the demand forecast. BD_POISSON_WKS	0	0 - 6
Alpha: Poisson. Smoothing parameter used for Poisson demand strategy. This parameter determines how quickly the weights decay for past weeks in-season data. BD_POISSON_PARAM	0.5	0–1
Demand Interval - Log Linear. Number of in-season weeks to be used for demand forecasting using Log linear demand strategy.0 implies use entire history from current season to generate the demand forecast. BD_EST_LOG_LINEAR	4	0 - 6
Alpha - Log Linear. Smoothing parameter used for Log Linear demand strategy. This parameter determines how quickly the weights decay for past weeks in-season data. BD_LOG_LINEAR_PARAM	0.7	0 - 1
Base demand - Sales units Threshold. Threshold used for determining the base demand strategy to use for an item. Items with sales lower than the threshold will use Poisson demand strategy other item will use log linear demand strategy. BD_LL_SLS_THRESH_QTY	20	15 - 30
Base demand - weeks used for Threshold. Number of weeks of data used for calculating the Base demand - Sales units threshold.BD_AE_SLS_LEN_WKS	4	1 - 5

Table 13-21 (Cont.) Demand Forecasting

Name and Description	Default Value	Range of Values
Store weights - Duration of data. Number of weeks of data used for calculation of store weights.BD_SW_SLS_LEN_WKS	13	13 - 52

New Stores

Retailers regularly open or close stores in different markets. The base demand forecasting process above accounts for stores that are operational. However, sometimes, retailers plan for new stores and would like the forecasting process to account for the additional increase (or decrease) in demand that will be generated due to the planned store opening (or closure). For example, a retailer might be planning to open a new store ten weeks in the future and would like to account for this in the demand forecasting process. LPO provides this functionality in which the retailer must provide a like store through the RSE_LIKE_LOC_STG interface. Once this mapping is provided, the LPO forecasting module adjusts the demand accordingly and the optimization considers the additional increase (or decrease) in demand for the new store (or closed store). Once the new store is operational, the sister store sales potential is discarded and it switches to the store's base demand forecasting process. This functionality helps retailers appropriately forecast demand based on planned store openings or closures. The following two examples illustrate the functionality.

Example 1: Sales at the Beginning of Week 2

Table 13-22 Sales at the Beginning of Week 2

Store	Week 1	Week 2	Week 3	Week 4	Week 5	Total Season
Store 1	100	120	100	120	100	540
Store 2	100	120	100	120	100	540
Store 3			Opens 50	50	80	180
Store 4		Opens 40	50	60	80	230
Total Sales	200	280	300	350	360	1490

Example 2: Sales at the Beginning of Week 4

Table 13-23 Sales at the Beginning of Week 4

Store	Week 1	Week 2	Week 3	Week 4	Week 5	Total Season
Store 1	150	150	140	140	140	720
Store 2	100	120	100	120	100	540
Store 3			Opened 100	100	160	360

Table 13-23 (Cont.) Sales at the Beginning of Week 4

Store	Week 1	Week 2	Week 3	Week 4	Week 5	Total Season
Store 4		Opened 40	50	60	80	230
Total Sales	250	310	390	420	480	1810

Optimization Science

This section describes the business rules that are available and the science behind the optimization algorithms used in the LPO application. It does not provide all the details of the algorithm. However, it does provide some guidance so that you can troubleshoot and resolve issues quickly during an implementation.

Business Rules

The implementation of LPO is determined by a retailer's individual business requirements. A complete and accurate configuration of business rules is essential to the accuracy and success of the recommendations and forecasts.

Some of the business requirements or rules include:

- The retailer's pricing strategy for promotions, markdowns, and targeted offers. Does the retailer want to maximize the revenue or profit margin?
- The level of the merchandise hierarchy and of the location hierarchy at which the sales data will be provided.
- The day of the week that markdowns are effective. Typical start and end date for promotions during a week.
- Whether markdowns are effective on the same day or different days for all departments.
- Whether promotions, budget, and distribution center data will be included in the data feed.
- How eligible items (items to be processed by LPO in a given week) are defined (for example, if an item exists in the most recent sales data feed and has a threshold number of units on hand).
- The kinds of inventory that must be considered by the batch process for optimization.

It is essential that considerable time be spent on understanding the business rules so that the global defaults can be populated in RSE_CONFIG with APPL_CODE = 'PRO' and so most of the departments will not require further tweaking of the batch runs or for ad hoc runs. The configuration that must be created in conjunction with the business requirements are:

Table 13-24 Business Rules

PARAM_NAME	PARAM_VALUE	DESCR
DEFAULT_APPL_USER	OO_BATCH_USR	User identifier to be used for batch activities that require user tracking.

Table 13-24 (Cont.) Business Rules

PARAM_NAME	PARAM_VALUE	DESCR
PRO_PROD_HIER_TYPE	3	The hierarchy ID to use for the product (Installation configuration).
PRO_LOC_HIER_TYPE	2	The hierarchy ID to use for the location (Installation configuration).
PRO_CAL_HIER_TYPE	11	The hierarchy ID to use for the calendar (Installation configuration).
PRO_CUSTSEG_HIER_TYPE	4	The hierarchy ID to use for the customer segments (Installation configuration).
PRO_CAL_HIER_PROCESSING _LVL	4	The calendar hierarchy level at which PRO will define RUNs for optimization.
PRO_PROD_HIER_RUN_SETU P_LVL	4	The merchandise hierarchy level at which PRO will setup/create RUNs.
PRO_PROD_HIER_PROCESSIN G_LVL	5	The merchandise hierarchy level at which PRO will optimize RUNs.
PRO_LOC_HIER_PROCESSING _LVL	2	The location hierarchy level at which PRO will define RUNs for optimization.
PRO_CUST_HIER_PROCESSIN G_LVL	2	Default customer segment level at which PRO will define RUNs for optimization.
PRO_OPT_LOC_REC_LVL	2	Default location level at which price recommendations will be generated.
PRO_OPT_CUST_REC_LVL	1	Default customer segment level at which price recommendations will be generated (whole population).
PRO_OPT_TIME_REC_LVL	4	Default calendar level at which price recommendations will be generated (week).
PRO_OPT_MERCH_REC_LVL	9	Default merchandise level at which price recommendations will be generated (STYLE-COLOR).
PRO_OPT_MECH_REC	NONE	Default targeted offer mechanics for which price recommendations will be generated.
PRO_OPT_MKTG_REC	NONE	Default targeted offer marketing aspect for which price recommendations will be generated.
PRO_SALVAGE_VALUE	0	Salvage value is the value of the product after the season ends. default value is 0.

Table 13-24 (Cont.) Business Rules

PARAM_NAME	PARAM_VALUE	DESCR
PRO_TR_NO_TOUCH_AFTER_LANDING	0	Default no touch after landing (2 weeks).
PRO_TR_LENGTH_OF_PROMOTIONS	0.6	Default length of promotion as a percentage of the whole season.
PRO_TR_MAX_LENGTH_OF_PROMOTION	1	Default maximum length (weeks) of a promotion.
PRO_TR_LENGTH_OF_MKDN	0.4	Default length of markdown as a percentage of the whole season.
PRO_TR_NO_TOUCH_END_OF_LIFE	0	Default no touch at the end of life (1.5 weeks).
PRO_PR_FIRST_PROMO_MIN_DISC_PCT	0	Default minimum discount percentage for the first promotion.
PRO_PR_FIRST_PROMO_MAX_DISC_PCT	1	Default maximum discount percentage for the first promotion.
PRO_PR_OTHER_PROMO_MIN_DISC_PCT	0	Default minimum discount percentage for promotions other than the first one.
PRO_PR_OTHER_PROMO_MAX_DISC_PCT	1	Default maximum discount percentage for promotions other than the first one.
PRO_PR_MIN_TIME_BETWEEN_PROMOS	1	Default minimum time separation between any two consecutive promotions (1 week).
PRO_PR_PROMO_START_DAY	MONDAY	Day of the week promotions start
PRO_PR_PROMO_END_DAY	SUNDAY	Day of the week promotions end
PRO_PR_DAY1_WGT	0.14	Promotion weight for day 1 of the week
PRO_PR_DAY2_WGT	0.14	Promotion weight for day 2 of the week
PRO_PR_DAY3_WGT	0.14	Promotion weight for day 3 of the week
PRO_PR_DAY4_WGT	0.14	Promotion weight for day 4 of the week
PRO_PR_DAY5_WGT	0.14	Promotion weight for day 5 of the week
PRO_PR_DAY6_WGT	0.15	Promotion weight for day 6 of the week
PRO_PR_DAY7_WGT	0.15	Promotion weight for day 7 of the week
PRO_MR_FIRST_MKDN_MIN_DISC_PCT	0	Default minimum discount percentage for the first markdown.
PRO_MR_FIRST_MKDN_MAX_DISC_PCT	1	Default maximum discount percentage for the first markdown.

Table 13-24 (Cont.) Business Rules

PARAM_NAME	PARAM_VALUE	DESCR
PRO_MR_OTHER_MKDN_MIN_DISC_PCT	0	Default minimum discount percentage for markdowns other than the first one.
PRO_MR_OTHER_MKDN_MAX_DISC_PCT	1	Default maximum discount percentage for markdowns other than the first one.
PRO_MR_MIN_TIME_BETWEEN_MKDN	1	Default minimum time separation between any two consecutive markdowns (1 week).
PRO_MR_MKDN_START_DAY	MONDAY	Day of the week markdown start
PRO_MR_MKDN_END_DAY	SUNDAY	Day of the week markdown end
PRO_MR_DAY1_WGT	0.14	Markdown weight for day 1 of the week
PRO_MR_DAY2_WGT	0.14	Markdown weight for day 2 of the week
PRO_MR_DAY3_WGT	0.14	Markdown weight for day 3 of the week
PRO_MR_DAY4_WGT	0.14	Markdown weight for day 4 of the week
PRO_MR_DAY5_WGT	0.14	Markdown weight for day 5 of the week
PRO_MR_DAY6_WGT	0.15	Markdown weight for day 6 of the week
PRO_MR_DAY7_WGT	0.15	Markdown weight for day 7 of the week
PRO_ST_END_REGULAR_SEASON	0.15	Default percentage of sell-through for each individual product at end of regular periods.
PRO_ST_END_CLEARANCE_SEASON	0.85	Default percentage of sell-through for each individual product at end of clearance season.
PRO_ST_TGT_TYPE	HARD	Default sell-through target type for each individual product.
PRO_ST_HARD_TGT_FLG	Y	Default value for sell-through hard target flag.
PRO_DFLT_RETURN_PCT	0.01	Default percentage of sales returned.
PRO_DFLT_RETURN_LEAD_TIME	2	Default return lead time in weeks (time when the return happens after the purchase is made).
PRO_PR_MIN_COST	Y	Promotions recommendations cannot be lower than the cost of the product.
PRO_MR_MIN_COST	Y	Markdown recommendations cannot be lower than the cost of the product.

Table 13-24 (Cont.) Business Rules

PARAM_NAME	PARAM_VALUE	DESCR
PRO_BASE_INITIAL_BUDGET	9999999	Initial Budget for Base Scenarios.
PRO_SEAS_CURVE_WEEKS_S WITCH	42	Seasonality Curve.
PRO_OFFERMAX	3	Maximum number of offers for segment in a given time period.
PRO_LOWRR	0.20	threshold value used to pick at least one low redemption rate offer for a segment and given time period.
PRO_HIGHRR	0.80	High Redemption Rate. This is a threshold value used to pick at least one high redemption offer for a given segment and given time period.
PRO_MAX_OFFERS_PER_SEG MENT	3	This is the number of top offers that will be selected per customer_segment/class.
PRO_INV_QTY_BOH	Y	Specifies if the inventory on hand must be included into the total inventory.
PRO_INV_QTY_ON_ORD	Y	Specifies if the inventory on order must be included into the total inventory.
PRO_INV_QTY_IN_TRANSIT	Y	Specifies if the inventory in transit must be included into the total inventory.
PRO_USE_ABSOLUTE_CAL_FL G	Y	Y/N Indicator. Identifies if the absolute calendar must be used for Temporal Rules.
PRO_USE_LIFECYCLE_FATIGU E	N	Y/N Indicator. Identifies if the life cycle fatigue must be enabled or not.

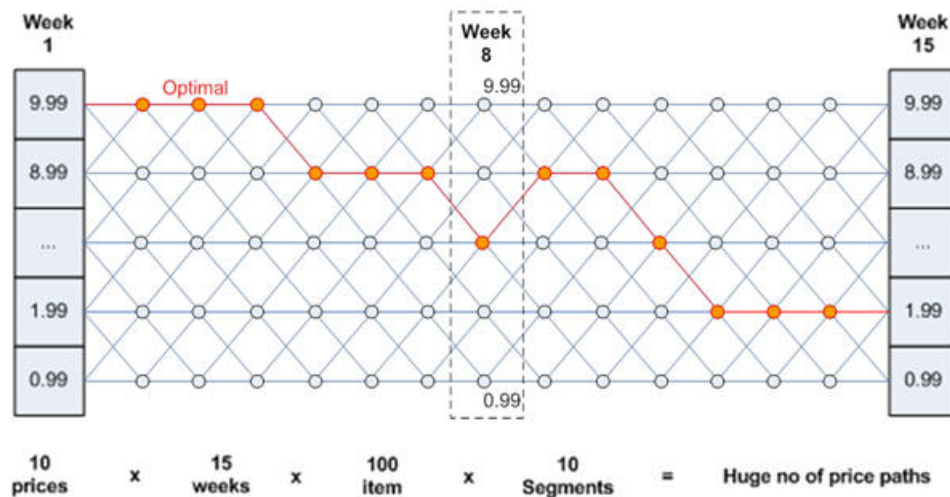
Optimization Algorithm Overview

The optimization algorithm analyzes trade-offs between a set of available price paths and picks the best price path. It then informs the user when to promote, how deep to promote, when to mark down, how deep to mark down, when to provide a targeted offer, and how deep the targeted offer must be. Note that the Targeted Offers are aligned to be in the same period when the promotion occurs.

The algorithm uses sophisticated mathematical modeling techniques to analyze all possible solutions to generate the best possible solution. The optimization algorithm is provided an objective (for example, maximize total revenue over all items in the season), business rules or restrictions, and the demand parameters for a particular item. The algorithm analyzes the trade-offs between all possible solutions (see [Figure 13-6](#)) and picks the solution that provides the best value for the objective. All the restrictions imposed by the user are treated as required; that is, all possible solutions must satisfy that particular criterion. The only constraint that can be enforced as soft is the sell-through target constraint. The user can make this constraint soft,

which tells the optimization that this constraint does not have to be met. Optimization will try to satisfy this constraint, and, if it cannot, it will not return an infeasible or no solution.

Figure 13-6 Optimization Algorithm



To provide a sense of the complexity of the optimization, here is a sample of the trade-offs that are analyzed.

- Is it better to provide a promotion early in the life or wait until later in life to mark down? Will conducting a promotion now help me given a shallower markdown at a later point in life?
- There are planned promotions scheduled at different points in time. Can these promotions be used and figure out whether an item needs further promotions or markdowns?
- Is providing a promotion at a particular week useful? Will it help meet the sell-through target? Will it increase revenue?
- There are customer segments with different response to price cuts. How can targeted offer provide a promotion that will entice the customer in customer segment A?
- Does providing a promotion or markdown or targeted offer for an item cause a conflict with the imposed business rules? For example, a user might determine that the item cannot be given more than 30% discount.

Model Apply

Forecasting is applied in the optimization using the demand parameters supplied. Suppose the user wants to generate weekly price recommendations. Depending on the day of the week when markdown is effective, forecasting is adjusted using daily weights to reflect that there are two prices in effect for that week. The same logic applies for promotions as well.

Objective Function

The objective function specified by the user plays a major role in determining which solutions are considered best. For example, if the user specifies the objective function as maximize profit margin, then the algorithm will look for solutions that are superior on profit margin and not necessarily on the other KPIs such as revenue and sales volume. Sometimes, understanding why an item got such a price recommendation might be as trivial as looking at the objective function contribution of that particular item to the objective function.

Constraints

If the objective function focuses on the best possible solution, then constraints work in the opposite direction, by restricting the set of possible solutions. For example, if the objective function says to select the most profitable discount for an item A, a constraint on item A may say it is not possible to have more than certain discount percentage.

Optimization enforces all constraints as required; that is, it finds all the solutions that satisfy all the constraints that are specified by the user. Sometimes, inadvertently, the user might specify conflicting constraints that can result in no solution or unexpected solutions. Often, a resolution can be found by just understanding the implications of individual constraints. More often than not, the user must analyze the interplay between two or more constraints to understand the solution. `PRO_RUN_SANITY_CHECK_RSE_VW` contains information on the errors/alerts/warnings generated.

LPO supports a variety of constraints, and it is essential for the user to understand the purpose and role of each constraint in the optimization.

Control and Tactical Center

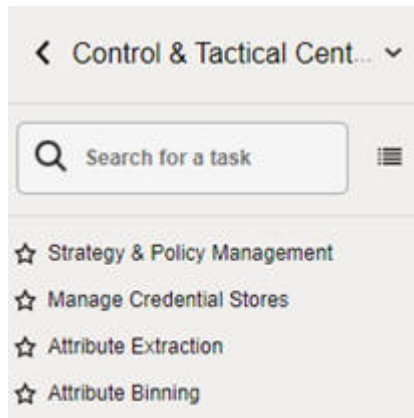
This chapter describes the Control and Tactical Center, where an administrative user can access and manage configurations for different applications.

Under the Control and Tactical Center, a user can access Strategy and Policy Management, which is the central place for managing the configurations of different applications and for setting up and configuring the demand forecasting runs. In addition, the modules for managing product attributes (Attribute Binning) as well as the link for managing web service-related credentials can be also found under the Control and Tactical Center.

Depending on the role, a user will see one or multiple of the following links, as shown below. Typically, the administrative user has access to all of the following.

- Strategy and Policy Management
- Manage Credential Stores
- Attribute Binning

Figure 14-1 Control and Tactical Center



Strategy and Policy Management

In Strategy and Policy Management, a user can edit the configurations of different applications using the Manage System Configurations screen. Only the tables that are editable can be accessed in this screen. Within each table, one or multiple columns are editable and a user can override the values in those columns.

Manage Forecast Configurations can be used to set up, manage, and configure the demand forecasting runs for different applications such as Lifecycle Pricing Optimization (LPO), Inventory Planning Optimization (IPO), Inventory Planning Optimization-Demand Forecasting (IPO-DF), Inventory Planning Optimization-Lifecycle Allocation and Replenishment (IPO-LAR), Assortment Planning (AP), and Merchandise Financial Planning (MFP).

Forecast Configuration for MFP and AP

To configure the forecast process for MFP and AP, complete the following two steps:

1. Use the Manage System Configurations screen to review and modify the global configurations in RSE_CONFIG. For further details, see "[Configuration Updates](#)". The relevant configurations in RSE_CONFIG that must be reviewed and edited as required are listed in the following table.

Table 14-1 Configurations

APPL CODE	PARAM NAME	PARAM VALUE	DESCR
RSE	EXTENDED_HIERARCHY_SRC	Default value is NON-RMS.	Data source providing extended hierarchy data RMS/NON-RMS.
RSE	LOAD_EXTENDED_PROD_HIER	Default value is N. If the product hierarchy data had 9 levels, set this value as Y. If it has 7 levels, keep this value as N.	Y/N Value. This parameter is used by the product hierarchy ETL to determine if the extended product hierarchy is required.
PMO	PMO_PROD_HIER_TYPE	Default value is 3. If the product hierarchy data has nine levels (i.e., it has an extended hierarchy), keep this value as 3. If it has seven levels, change this value to 1.	The hierarchy ID to use for the product (installation configuration).
RSE	PROD_HIER_SLSTXN_HIER_LEV EL_ID	Default value is 9.	This parameter identifies the hierarchy level at which sales transactions are provided (7-Style, 8-Style/color, or 9 Style/color/Size). It <i>must</i> match the extended hierarchy leaf level.
RSE	FCST_PURGE_RETENTION_DAYS	Default value is 180.	Number of days to wait before permanently deleting run data.
RSE	FCST_PURGE_ALL_RUNS	Default value is N.	Y/N flag indicating all run data should be deleted.
RSE	FCST_TY_PURGE_RETENTION_DAYS	Default value is 180.	Number of days to wait before permanently deleting run type data.
RSE	FCST_TY_PURGE_ALL_RUNS	Default value is N.	Y/N flag indicating all run type data should be deleted.

Table 14-1 (Cont.) Configurations

APPL CODE	PARAM NAME	PARAM VALUE	DESCR
RSE	SWITCH_TO_TXN_LOAD_DATE	Default value is SYSDATE.	For customers who provide FACT data at aggregate level, this date indicates the stop point for loading aggregate data. If the SYSDATE is greater than or equal to this date, the data load from RI FACT tables to AIF will not run. Applicable date format YYYYMMDD.
PMO	PMO_AGGR_INVENTORY_DATA_FLG	Default value is Y.	Specifies if inventory data is present and if it should be used when aggregating activities data. Set this value to N if inventory data is not loaded (inventory data is not required for MFP and AP applications but it may be required for other applications such as LPO and IPO).

2. Use the Manage Forecast Configurations screen to set up the forecast runs for MFP and AP, as follows.

In Manage Forecast Configurations, start by setting up a run type in the Setup train stop. Click the + icon above the table and fill in the fields in the pop-up. For MFP and AP forecasting, the preferred forecast method is Automatic Exponential Smoothing. However, the Sales & Promo and Life Cycle forecast methods are also supported. You must create a run type for each forecast measure/forecast intersection combination that is required for MFP and/or AP.

Once you are done with setting up the run types, click **Start Data Aggregation** in the Setup train stop. Select all the run types that were created and click **Submit**. When the aggregation is complete, the aggregation status will change to Complete. At this point, ad-hoc test runs and batch runs can be created and executed to generate a forecast.

To create an ad-hoc run, go to the **Test** train stop. First, click a run type in the top table and then click the + icon in the bottom table to create a run. In the Create Run pop-up, you can change the configurations parameters related to estimation process and forecast process in their respective tabs. For example, if you want to test a run using the Bayesian method, edit the Estimation Method parameter in the Estimation tab using the **Edit** icon above the table. After modifying and reviewing the configuration parameters, click **Submit** to start the run. Once the run is complete, the status will change to Forecast Generation Complete.

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

Note

If the run type is active, you will only be able to view the parameters. In order to edit the parameters, the run type must be inactive.

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

Run Type Configurations for MFP and AP to Set Up GA Runs

To set up GA runs, create one run type for each forecast intersection/measure combination using the Automatic Exponential Smoothing method. The following table shows the configurations for each run type.

Table 14-2 Run Type Configurations

	Forecast Level (Merchandise/ Location/Calendar)	Forecast Measure	Forecast Method	Data Source
Location Plan (MFP)	Department/Location/ Week	Total Gross Sales Units	Automatic Exponential Smoothing for all run types	Store Sales for all run types
		Total Gross Sales Amount		
		Total Returns Units		
		Total Returns Amount		
Location Target (MFP)	Company/Location/ Week	Total Gross Sales Units		
		Total Gross Sales Amount		
		Total Returns Units		
		Total Returns Amount		
Merchandise Plan (MFP)	Subclass/Area/Week	Clearance Gross Sales Units		
		Clearance Gross Sales Amount		
		Regular and Promotion Gross Sales Units		
		Regular and Promotion Gross Sales Amount		

Table 14-2 (Cont.) Run Type Configurations

	Forecast Level (Merchandise/ Location/Calendar)	Forecast Measure	Forecast Method	Data Source
Merchandise Target (MFP)	Department/Area/Week	Total Returns Units		
		Total Returns Amount		
		Other Gross Sales Units		
		Other Gross Sales Amount		
		Clearance Gross Sales Units		
		Clearance Gross Sales Amount		
		Regular and Promotion Gross Sales Units		
		Regular and Promotion Gross Sales Amount		
		Total Returns Units		
		Total Returns Amount		
		Other Gross Sales Units		
		Other Gross Sales Amount		
Assortment Planning (AP)	Subclass/Location/ Week	Clearance Gross Sales Unit		
		Clearance Gross Sales Amount		
		Regular and Promotion Gross Sales Unit		
		Regular and Promotion Gross Sales Amount		
Item Planning (AP)	SKU/Location/Week	Clearance Gross Sales Units		

Table 14-2 (Cont.) Run Type Configurations

Forecast Level (Merchandise/ Location/Calendar)	Forecast Measure	Forecast Method	Data Source
	Clearance Gross Sales Amount		
	Regular and Promotion Gross Sales Units		
	Regular and Promotion Gross Sales Amount		

Note

For all batch runs, in addition to the pre-season forecast, the Bayesian forecast is also generated by default. In the Manage train stop, you must only set the method for the pre-season forecast (either as Automatic Exponential Smoothing or Seasonal Exponential Smoothing).

Batch and Ad Hoc Jobs for MFP and AP Forecasting

The Batch and Ad Hoc jobs listed in the following table are used for loading foundation data (product hierarchy, location hierarchy, calendar, product attributes, and so on).

Table 14-3 Configuration and Main Data Load Jobs

JobName	Description	RmsBatch
ORASE_START_BATCH_JOB	ORASE_START_BATCH_JOB	rse_process_state_update.ksh
ORASE_START_BATCH_SET_ACTIVE_JOB	ORASE_START_BATCH_SET_ACTIVE_JOB	rse_batch_type_active.ksh
ORASE_START_BATCH_REFRESH_RESTRICTION_JOB	ORASE_START_BATCH_REFRESH_RESTRICTION_JOB	rse_batch_freq_restriction.ksh
ORASE_START_BATCH_END_JOB	ORASE_START_BATCH_END_JOB	rse_process_state_update.ksh
RSE_WEEKLY_INPUT_FILES_START_JOB	RSE_WEEKLY_INPUT_FILES_START_JOB	rse_process_state_update.ksh
WEEKLY_INPUT_FILES_WAIT_JOB	WEEKLY_INPUT_FILES_WAIT_JOB	rse_batch_zip_file_wait.ksh
WEEKLY_INPUT_FILES_VALIDATE_JOB	WEEKLY_INPUT_FILES_VALIDATE_JOB	rse_batch_zip_file_extract.ksh
WEEKLY_INPUT_FILES_COPY_JOB	WEEKLY_INPUT_FILES_COPY_JOB	rse_batch_inbound_file_copy.ksh

Table 14-3 (Cont.) Configuration and Main Data Load Jobs

JobName	Description	RmsBatch
RSE_WEEKLY_INPUT_FILES_END_JOB	RSE_WEEKLY_INPUT_FILES_END_JOB	rse_process_state_update.ksh
RSE_PROD_SRC_XREF_LOAD_JOB	RSE_PROD_SRC_XREF_LOAD_JOB	rse_prod_src_xref_load.ksh
RSE_PROD_HIER_LOAD_JOB	RSE_PROD_HIER_LOAD_JOB	rse_prod_hier_load.ksh
RSE_PROD_TC_LOAD_JOB	RSE_PROD_TC_LOAD_JOB	rse_prod_tc_load.ksh
RSE_PROD_DH_LOAD_JOB	RSE_PROD_DH_LOAD_JOB	rse_prod_dh_load.ksh
RSE_PROD_GROUP_LOAD_JOB	RSE_PROD_GROUP_LOAD_JOB	rse_load_prod_group.ksh
RSE_LOC_SRC_XREF_LOAD_JOB	RSE_LOC_SRC_XREF_LOAD_JOB	rse_loc_src_xref_load.ksh
RSE_LOC_HIER_LOAD_JOB	RSE_LOC_HIER_LOAD_JOB	rse_loc_hier_load.ksh
RSE_LOC_HIER_TC_LOAD_JOB	RSE_LOC_HIER_TC_LOAD_JOB	rse_loc_hier_tc_load.ksh
RSE_LOC_HIER_DH_LOAD_JOB	RSE_LOC_HIER_DH_LOAD_JOB	rse_loc_hier_dh_load.ksh
RSE_CDA_ETL_LOAD_LOC_JOB	RSE_CDA_ETL_LOAD_LOC_JOB	rse_cda_etl_load_location.ksh
RSE_LIKE_LOC_STG_JOB	RSE_LIKE_LOC_STG_JOB	rse_like_loc_stg.ksh
RSE_LIKE_LOC_COPY_JOB	RSE_LIKE_LOC_COPY_JOB	
RSE_LIKE_LOC_STG_CNE_JOB	RSE_LIKE_LOC_STG_CNE_JOB	
RSE_LIKE_LOC_LOAD_JOB	RSE_LIKE_LOC_LOAD_JOB	rse_like_loc_load.ksh
RSE_LIKE_PROD_LOAD_START_JOB	RSE_LIKE_PROD_LOAD_START_JOB	rse_process_state_update.ksh
RSE_LIKE_PROD_STG_JOB	RSE_LIKE_PROD_STG_JOB	rse_like_prod_stg.ksh
RSE_LIKE_PROD_COPY_JOB	RSE_LIKE_PROD_COPY_JOB	
RSE_LIKE_PROD_STG_CNE_JOB	RSE_LIKE_PROD_STG_CNE_JOB	
RSE_LIKE_PROD_LOAD_JOB	RSE_LIKE_PROD_LOAD_JOB	rse_like_prod_load.ksh
RSE_LIKE_PROD_LOAD_END_JOB	RSE_LIKE_PROD_LOAD_END_JOB	rse_process_state_update.ksh
RSE_MD_CDA_VALUES_STG_JOB	RSE_MD_CDA_VALUES_STG_JOB	rse_md_cda_values_stg.ksh
RSE_MD_CDA_VALUES_COPY_JOB	RSE_MD_CDA_VALUES_COPY_JOB	
RSE_MD_CDA_VALUES_STG_CNE_JOB	RSE_MD_CDA_VALUES_STG_CNE_JOB	
RSE_PR_LC_CDA_STG_JOB	RSE_PR_LC_CDA_STG_JOB	rse_pr_lc_cda_stg.ksh
RSE_PR_LC_CDA_COPY_JOB	RSE_PR_LC_CDA_COPY_JOB	
RSE_PR_LC_CDA_STG_CNE_JOB	RSE_PR_LC_CDA_STG_CNE_JOB	
RSE_PR_LC_CAL_CDA_JOB	RSE_PR_LC_CAL_CDA_JOB	rse_pr_lc_cal_cda_stg.ksh

Table 14-3 (Cont.) Configuration and Main Data Load Jobs

JobName	Description	RmsBatch
RSE_PR_LC_CAL_CDA_COPY_JOB	RSE_PR_LC_CAL_CDA_COPY_JOB	
RSE_PR_LC_CAL_CDA_STG_CNE_JOB	RSE_PR_LC_CAL_CDA_STG_CNE_JOB	
RSE_MD_CDA_STG_JOB	RSE_MD_CDA_STG_JOB	rse_md_cda_stg.ksh
RSE_MD_CDA_COPY_JOB	RSE_MD_CDA_COPY_JOB	
RSE_MD_CDA_STG_CNE_JOB	RSE_MD_CDA_STG_CNE_JOB	
RSE_MD_CDA_LOAD_JOB	RSE_MD_CDA_LOAD_JOB	rse_md_cda_load.ksh
RSE_MD_CDA_VALUES_LOAD_JOB	RSE_MD_CDA_VALUES_LOAD_JOB	rse_md_cda_values_load.ksh
RSE_DATA_STAGING_END_JOB	RSE_DATA_STAGING_END_JOB	rse_process_state_update.ksh
RSE_PROD_ATTR_LOAD_STAR_T_JOB	RSE_PROD_ATTR_LOAD_STAR_T_JOB	rse_process_state_update.ksh
RSE_CDA_ETL_LOAD_PROD_JOB	RSE_CDA_ETL_LOAD_PROD_JOB	rse_cda_etl_load_product.ksh
RSE_PROD_ATTR_GRP_VALUE_STG_JOB	RSE_PROD_ATTR_GRP_VALUE_STG_JOB	rse_prod_attr_grp_value_stg.ksh
RSE_PROD_ATTR_GRP_VALUE_COPY_JOB	RSE_PROD_ATTR_GRP_VALUE_COPY_JOB	
RSE_PROD_ATTR_GRP_VALUE_STG_CNE_JOB	RSE_PROD_ATTR_GRP_VALUE_STG_CNE_JOB	
RSE_PROD_ATTR_GRP_VALUE_LOAD_JOB	RSE_PROD_ATTR_GRP_VALUE_LOAD_JOB	rse_prod_attr_grp_value_load.ksh
RSE_PROD_ATTR_VALUE_XREF_STG_JOB	RSE_PROD_ATTR_VALUE_XREF_STG_JOB	rse_prod_attr_value_xref_stg.ksh
RSE_PROD_ATTR_VALUE_XREF_COPY_JOB	RSE_PROD_ATTR_VALUE_XREF_COPY_JOB	
RSE_PROD_ATTR_VALUE_XREF_STG_CNE_JOB	RSE_PROD_ATTR_VALUE_XREF_STG_CNE_JOB	
RSE_PROD_ATTR_VALUE_XREF_LOAD_JOB	RSE_PROD_ATTR_VALUE_XREF_LOAD_JOB	rse_prod_attr_value_xref_load.ksh
RSE_CM_GRP_XREF_LOAD_JOB	RSE_CM_GRP_XREF_LOAD_JOB	rse_cm_grp_xref_load.ksh
RSE_CM_GRP_HIER_LOAD_JOB	RSE_CM_GRP_HIER_LOAD_JOB	rse_cm_grp_hier_load.ksh
RSE_CM_GRP_TC_LOAD_JOB	RSE_CM_GRP_TC_LOAD_JOB	rse_cm_grp_tc_load.ksh
RSE_CM_GRP_DH_LOAD_JOB	RSE_CM_GRP_DH_LOAD_JOB	rse_cm_grp_dh_load.ksh
RSE_TRADE_AREA_SRC_XREF_LOAD_JOB	RSE_TRADE_AREA_SRC_XREF_LOAD_JOB	rse_trade_area_src_xref_load.ksh
RSE_TRADE_AREA_HIER_LOAD_JOB	RSE_TRADE_AREA_HIER_LOAD_JOB	rse_trade_area_hier_load.ksh
RSE_TRADE_AREA_TC_LOAD_JOB	RSE_TRADE_AREA_TC_LOAD_JOB	rse_trade_area_tc_load.ksh

Table 14-3 (Cont.) Configuration and Main Data Load Jobs

JobName	Description	RmsBatch
RSE_TRADE_AREA_DH_LOAD_JOB	RSE_TRADE_AREA_DH_LOAD_JOB	rse_trade_area_dh_load.ksh
RSE_CUSTSEG_SRC_XREF_LOAD_JOB	RSE_CUSTSEG_SRC_XREF_LOAD_JOB	rse_custseg_src_xref_load.ksh
RSE_CUSTSEG_HIER_LOAD_JOB	RSE_CUSTSEG_HIER_LOAD_JOB	rse_custseg_hier_load.ksh
RSE_CUSTSEG_HIER_TC_LOAD_JOB	RSE_CUSTSEG_HIER_TC_LOAD_JOB	rse_custseg_hier_tc_load.ksh
RSE_CUSTSEG_CUST_XREF_LOAD_JOB	RSE_CUSTSEG_CUST_XREF_LOAD_JOB	rse_custseg_cust_xref_load.ksh
RSE_CONSEG_LOAD_JOB	RSE_CONSEG_LOAD_JOB	rse_conseg_load.ksh
RSE_CONSEG_ALLOC_LOAD_JOB	RSE_CONSEG_ALLOC_LOAD_JOB	rse_conseg_alloc_load.ksh
RSE_CUSTSEG_ALLOC_LOAD_JOB	RSE_CUSTSEG_ALLOC_LOAD_JOB	rse_custseg_alloc_load.ksh
RSE_REGULAR_MAIN_LOAD_JOB	RSE_REGULAR_MAIN_LOAD_JOB	rse_regular_main_load.ksh
RSE_FISCAL_MAIN_LOAD_JOB	RSE_FISCAL_MAIN_LOAD_JOB	rse_fiscal_main_load.ksh
RSE_DIMENSION_LOAD_END_START_JOB	RSE_DIMENSION_LOAD_END_START_JOB	rse_process_state_update.ksh
RSE_DIMENSION_LOAD_END_END_JOB	RSE_DIMENSION_LOAD_END_END_JOB	rse_process_state_update.ksh
RSE_DIMENSION_LOAD_START_START_JOB	RSE_DIMENSION_LOAD_START_START_JOB	rse_process_state_update.ksh
RSE_DIMENSION_LOAD_START_END_JOB	RSE_DIMENSION_LOAD_START_END_JOB	rse_process_state_update.ksh
RSE_SLS_START_START_JOB	RSE_SLS_START_START_JOB	rse_process_state_update.ksh
RSE_SLS_START_END_JOB	RSE_SLS_START_END_JOB	rse_process_state_update.ksh
RSE_SLS_END_START_JOB	RSE_SLS_END_START_JOB	rse_process_state_update.ksh
RSE_SLS_END_END_JOB	RSE_SLS_END_END_JOB	rse_process_state_update.ksh
RSE_SLS_TXN_SETUP_JOB	Populates the RSE_SLS_TXN table with sales transaction data.	rse_sls_txn_setup.ksh
RSE_SLS_TXN_PROCESS_JOB	Populates the RSE_SLS_TXN table with sales transaction data.	rse_sls_txn_process.ksh
RSE_WKLY_SLS_SETUP_JOB	Populates the RSE_SLS_PR_LC_WK table with sales data pulled from the RSE_SLS_TXN table and aggregated to the product/location/week level.	rse_wkly_sls_setup.ksh
RSE_WKLY_SLS_PROCESS_JOB	Populates the RSE_SLS_PR_LC_WK table with sales data pulled from the RSE_SLS_TXN table and aggregated to the product/location/week level.	rse_wkly_sls_process.ksh

Table 14-3 (Cont.) Configuration and Main Data Load Jobs

JobName	Description	RmsBatch
RSE_WKLY_SLS_SEG_SETUP_JOB	Populates the RSE_SLS_PR_LC_CS_WK table with sales data pulled from the RSE_SLS_TXN table and aggregated to the product/location/cust. segm./week level.	rse_wkly_sls_seg_setup.ksh
RSE_WKLY_SLS_SEG_PROCESS_JOB	Populates the RSE_SLS_PR_LC_CS_WK table with sales data pulled from the RSE_SLS_TXN table and aggregated to the product/location/cust. segm./week level.	rse_wkly_sls_seg_process.ksh
ORASE_END_START_JOB	ORASE_END_START_JOB	rse_process_state_update.ksh
ORASE_END_RUN_DATE_UPDATE_JOB	ORASE_END_RUN_DATE_UPDATE_JOB	rse_batch_run_date_update.ksh
ORASE_END_END_JOB	ORASE_END_END_JOB	rse_process_state_update.ksh
RSE_MASTER_ADHOC_JOB	Run RSE master script	rse_master.ksh
PMO_MASTER_ADHOC_JOB	Run PMO master script	pmo_master.ksh
PMO_CUMUL_SLS_SETUP_JOB	Setup job for cumulative sales calculation	pmo_cum_sls_load_setup.ksh
PMO_CUMUL_SLS_PROCESS_JOB	Process job for cumulative sales calculation	pmo_cum_sls_load_process.ksh
PMO_CUMUL_SLS_SETUP_ADHOC_JOB	Ad hoc setup job for cumulative sales calculation	pmo_cum_sls_load_setup.ksh
PMO_CUMUL_SLS_PROCESS_ADHOC_JOB	Ad hoc process job for cumulative sales calculation	pmo_cum_sls_load_process.ksh
PMO_ACTIVITY_STG_JOB	PMO Activity load setup job to aggregate sales data for store sales	pmo_activity_load_setup.ksh
PMO_ACTIVITY_LOAD_JOB	PMO Activity load process job to aggregate sales data for store sales	pmo_activity_load_process.ksh
PMO_ACTIVITY_STG_ADHOC_JOB	Ad hoc PMO Activity load setup job to aggregate sales data for store sales	pmo_activity_load_setup.ksh
PMO_ACTIVITY_LOAD_ADHOC_JOB	Ad hoc PMO Activity load process job to aggregate sales data for store sales	pmo_activity_load_process.ksh
PMO_ACTIVITY_WH_LOAD_SETUP_JOB	PMO Activity setup job to aggregate sales data for warehouse transfers	pmo_activity_wh_load_setup.ksh
PMO_ACTIVITY_WH_LOAD_PROCESS_JOB	PMO Activity process job to aggregate sales data for warehouse transfers	pmo_activity_wh_load_process.ksh
PMO_ACTIVITY_WH_LOAD_SETUP_ADHOC_JOB	Ad hoc PMO Activity setup job to aggregate sales data for warehouse transfers	pmo_activity_wh_load_setup.ksh
PMO_ACTIVITY_WH_LOAD_PROCESS_ADHOC_JOB	Ad hoc PMO Activity process job to aggregate sales data for warehouse transfers	pmo_activity_wh_load_process.ksh

Table 14-3 (Cont.) Configuration and Main Data Load Jobs

JobName	Description	RmsBatch
PMO_ACTIVITY_SHIPMENT_LOAD_SETUP_JOB	PMO Activity setup job to aggregate sales data for warehouse shipments	pmo_activity_shipment_load_setup.ksh
PMO_ACTIVITY_SHIPMENT_LOAD_PROCESS_JOB	PMO Activity process job to aggregate sales data for warehouse shipments	pmo_activity_shipment_load_process.ksh
PMO_ACTIVITY_SHIPMENT_LOAD_SETUP_ADHOC_JOB	Ad hoc PMO Activity setup job to aggregate sales data for warehouse shipments	pmo_activity_shipment_load_setup.ksh
PMO_ACTIVITY_SHIPMENT_LOAD_PROCESS_ADHOC_JOB	Ad hoc PMO Activity process job to aggregate sales data for warehouse shipments	pmo_activity_shipment_load_process.ksh
PMO_ACTIVITY_SALES_SHIP_LOAD_SETUP_JOB	PMO Activity setup job to aggregate sales data for store sales and warehouse shipments	pmo_activity_sales_ship_load_setup.ksh
PMO_ACTIVITY_SALES_SHIP_LOAD_PROCESS_JOB	PMO Activity process job to aggregate sales data for store sales and warehouse shipments	pmo_activity_sales_ship_load_process.ksh
PMO_ACTIVITY_SALES_SHIP_LOAD_SETUP_ADHOC_JOB	Ad hoc PMO Activity setup job to aggregate sales data for store sales and warehouse shipments	pmo_activity_sales_ship_load_setup.ksh
PMO_ACTIVITY_SALES_SHIP_LOAD_PROCESS_ADHOC_JOB	Ad hoc PMO Activity process job to aggregate sales data for store sales and warehouse shipments	pmo_activity_sales_ship_load_process.ksh
PMO_RETURN_DATAPREP_SETUP_ADHOC_JOB	Ad hoc PMO Returns data load preparation setup job	pmo_return_dataprep_setup.ksh
PMO_RETURN_DATAPREP_PROCESS_ADHOC_JOB	Ad hoc PMO Returns data load preparation processing job	pmo_return_dataprep_process.ksh
PMO_RETURN_CALC_PROCESS_ADHOC_JOB	Ad hoc PMO Returns calculations processing job	pmo_return_calculation.ksh
PMO_HOLIDAY_LOAD_ADHOC_JOB	Ad hoc PMO Holiday load job	pmo_holiday_load.ksh

The Batch and Ad hoc jobs listed in the following table are used to prepare weekly data and to run weekly batches for MFP and AP.

Table 14-4 Batch and Ad Hoc Jobs for MFP and AP Forecasting

JobName	Description	RmsBatch
RSE_FCST_SALES_PLAN_LOAD_JOB	Load job for sales plan data for Bayesian method	rse_fcst_sales_plan_load.ksh
RSE_ASSORT_PLAN_LOAD_STG_JOB	Loading assortment plan through interface for AP	rse_assort_plan_per_stg.ksh
RSE_ASSORT_PLAN_LOAD_COPY_JOB	Loading assortment plan through interface for AP	
RSE_ASSORT_PLAN_LOAD_STG_CNE_JOB	Loading assortment plan through interface for AP	
RSE_RDX_ASSORT_PLAN_IMPORT_JOB	Importing assortment plan from RDX to AIF for AP	rse_rdx_assort_plan_import.ksh

Table 14-4 (Cont.) Batch and Ad Hoc Jobs for MFP and AP Forecasting

JobName	Description	RmsBatch
RSE_ASSORT_PLAN_LOAD_JOB	Loading assortment plan through interface for AP	rse_assort_plan_per_load.ksh
RSE_PLANNING_PERIOD_STG_JOB	Loading assortment period through interface for AP	rse_planning_period_stg.ksh
RSE_PLANNING_PERIOD_COPY_JOB	Loading assortment period through interface for AP	
RSE_PLANNING_PERIOD_STG_CNE_JOB	Loading assortment period through interface for AP	
RSE_RDX_ASSORT_PERIOD_IMPORT_JOB	Importing assortment period from RDX to AIF for AP	rse_rdx_assort_period_import.ksh
RSE_PLANNING_PERIOD_LOAD_JOB	Loading assortment period through interface for AP	rse_planning_period_load.ksh
RSE_CREATE_MFP_BATCH_RUN_PROC_JOB	Create MFP and AP batch runs	rse_create_mfp_batch_run_proc.ksh
RSE_MFP_FCST_BATCH_PROCESS_JOB	Execute MFP and AP forecast batch	rse_fcst_mfp_batch_process.ksh
RSE_CREATE_MFP_BATCH_RUN_PROC_ADHOC_JOB	Ad hoc job to create MFP and AP batch runs	rse_create_mfp_batch_run_proc.ksh
RSE_MFP_FCST_BATCH_PROCESS_ADHOC_JOB	Ad hoc job to execute MFP and AP forecast batch	rse_fcst_mfp_batch_process.ksh

The forecast values generated by runs that are associated with active run types are exported to the RDX schema. The jobs for exporting the outputs to the RDX schema are listed in the following table.

Table 14-5 Export Jobs for MFP and AP from AIF to RDX

JobName	Description	RmsBatch
RSE_MFP_FCST_EXPORT_JOB	Export MFP forecast	rse_mfp_export.ksh
RSE_AP_FCST_EXPORT_JOB	Export AP forecast	rse_ap_export.ksh
RSE_MFP_FCST_EXPORT_ADHOC_JOB	Ad hoc job to export MFP forecast	rse_mfp_export.ksh
RSE_AP_FCST_EXPORT_ADHOC_JOB	Ad hoc job to export AP forecast	rse_ap_export.ksh
RSE_RDX_ASSORT_ELASTICITY_EXPORT_JOB	Export assortment elasticity for AP	rse_rdx_assort_elasticity_export.ksh
RSE_RDX_ASSORT_ELASTICITY_EXPORT_ADHOC_JOB	Ad hoc job to export assortment elasticity for AP	rse_rdx_assort_elasticity_export.ksh
RSE_RDX_LOC_CLUSTER_EXPORT_JOB	Export store clusters for AP	rse_rdx_loc_cluster_export.ksh
RSE_RDX_LOC_CLUSTER_EXPORT_ADHOC_JOB	Ad hoc job to export store clusters for AP	rse_rdx_loc_cluster_export.ksh
RSE_RDX_SIZE_PROFILE_EXPORT_JOB	Export size profiles for AP	rse_rdx_size_profile_export.ksh

Table 14-5 (Cont.) Export Jobs for MFP and AP from AIF to RDX

JobName	Description	RmsBatch
RSE_RDX_SIZE_PROFILE_EXP ORT_ADHOC_JOB	Ad hoc job to export size profiles for AP	rse_rdx_size_profile_export.ksh

Forecast Configuration for IPO (DF, IO, and LAR) and LPO

To configure forecast process for IPO and LPO, do the following.

1. Use the Manage System Configurations screen to review and modify the global configurations in RSE_CONFIG. For further details, see ["Configuration Updates"](#). The relevant configurations in RSE_CONFIG that must be reviewed and edited as required are listed in the following table.

Table 14-6 RSE Configurations

APPL_CODE	PARAM_NAME	PARAM_VALUE	DESCR
RSE	EXTENDED_HIERARCHY_SRC	Default value is NON-RMS.	Data source providing extended hierarchy data RMS/NON-RMS.
RSE	LOAD_EXTENDED_PRODUCT_HIER	Default value is N. If the product hierarchy data had 9 levels, set this value as Y. If it has 7 levels, keep this value as N.	Y/N Value. This parameter is used by the product hierarchy ETL to know if the extended product hierarchy is needed.
PMO	PMO_PROD_HIER_TYPE	Default value is 3. If the product hierarchy data has 9 levels (i.e., it has extended hierarchy), keep this value as 3. If it has 7 levels, change this value to 1.	The hierarchy id to use for the product (Installation configuration).
RSE	PROD_HIER_SLSTXN_HIER_LEVEL_ID	Default value is 9.	This parameter identifies the hierarchy level at which sales transactions are provided (7-Style, 8-Style/color or 9 Style/color/Size). It <i>must</i> match the extended hierarchy leaf level
PMO	PMO_AGGR_INVENTORY_DATA_FLG	Default value is Y.	Specifies if inventory data is present and if it should be used when aggregating activities data. Set this value to N if inventory data is not loaded (inventory data is not required for MFP and AP applications but it may be required for other applications such as LPO and IPO).

Table 14-6 (Cont.) RSE Configurations

APPL_CODE	PARAM_NAME	PARAM_VALUE	DESCR
RSE	SLS_TXN_EXCLUDE_LIA BILITY_SLS	Default value is N.	Y/N flag indicating if liability sales columns should be excluded when retrieving sales transaction data.
RSE	RSE_LOC_HIER_EXCL_ FRANCHISE	Default value is N.	Y/N flag indicating if franchise locations should be excluded from the location hierarchy.
RSE	PROMOTIONAL_SALES_ AVAIL	Default value is Y.	Y/N flag indicating if promotion data is available for Sales & Promo run types. Set this value to N if promotion data is not yet loaded into AIF.
RSE	DFLT_LIFE_CYCLE	Default value is SLC.	This parameter identifies the primary forecast method for the env. Keep this value as SLC if the primary forecast method is Life Cycle and change this value to LLC if the primary forecast method is Sales & Promo.
PMO	PMO_OUTLIER_CALCUL ATION_FLG	Default value is Y.	Y/N flag indicating if the Outlier flag calculation should happen internally in AIF.
RSE	FCST_PURGE_RETENTI ON_DAYS	Default value is 180.	Number of days to wait before permanently deleting run data.
RSE	FCST_PURGE_ALL_RUN S	Default value is N.	Y/N flag indicating all run data should be deleted.
RSE	FCST_TY_PURGE_RETE NTION_DAYS	Default value is 180.	Number of days to wait before permanently deleting run type data.
RSE	FCST_TY_PURGE_ALL_ RUNS	Default value is N.	Y/N flag indicating all run type data should be deleted.
RSE	RSE_FIRST_INV_RECEI PT_HIST_MAX_WEEKS	Default value is 106.	Records with inventory first receipt date older than these many weeks (from the latest week with inventory) will be excluded from model start date calculation.
RSE	INV_SRC_RI	Default value is Y.	Flag which indicates whether inventory is obtained from AIF data warehouse (Y) or loaded directly to AIF applications (N).

Table 14-6 (Cont.) RSE Configurations

APPL_CODE	PARAM_NAME	PARAM_VALUE	DESCR
RSE	INV_RC_DT_SRC_RI	Default value is Y.	Flag that indicates whether receipt dates are obtained from AIF data warehouse (Y) or loaded directly to AIF applications (N).
RSE	RSE_FCST_NEW_ITEM_STRATEGY	Default value is RDX.	Allowed values are RDX (for population from RDX), RI (for population from RI interface), and RSP (programmatic population using RSE_FCST_NEW_ITEM_AGE_THRESH).
RSE	RSE_FCST_NEW_ITEM_AGE_THRESH	Default value is 28.	When RSE_FCST_NEW_ITEM_STRATEGY is RSP, this threshold will be used to identify new items. If the number of days passed from the prod_created_dt of the item is smaller than or equal to this value, the item will be considered new.
RSE	RSE_BD_ENABLE_POISON	Default value is Y.	Y/N flag indicating whether to also use the Poisson method for calculating base demand for Sales & Promo run types.
RSE	SWITCH_TO_TXN_LOAD_DATE	Default value is SYSDATE.	For customers who provide FACT data at aggregate level, this date indicates the stop point for loading aggregate data. If the SYSDATE is greater than or equal to this date, the data load from RI FACT tables to AIF will not run. Applicable date format YYYYMMDD.
RSE	LOAD_HISTORICAL_HIERARCHY	Default value is Y.	Y/N Flag to indicate if the historical hierarchy needs to be saved.
RSE	HISTORICAL_AGGREGATION_FLG	Default value is N.	Y/N Flag indicating if the batch processing (aggregation, estimation, etc.) must use the historical product hierarchy including inactive products
RSE	APPR_FCST_EXPORT_NUM_WK	Default value is 4.	Number of weeks of approved forecast at week level export to RI/RMS.
RSE	APPR_FCST_DAY_EXPORT_NUM_DY	Default value is 28.	Number of days of approved forecast at day level export to RI/RMS.

2. Use the Manage Forecast Configurations screen to set up the forecast runs for IPO and LPO, as follows.

In Manage Forecast Configurations, start by setting up a run type in the Setup train stop. Click the + icon above the table and fill in the fields in the pop-up. For IPO forecasting, the preferred forecast method is Sales & Promo. For LPO forecasting, the preferred forecast methods are Life Cycle (while doing Promotion/Markdown Optimization) and Hierarchical Bayesian (while doing Regular Price Optimization). The customer should decide the primary forecast method for the environment and accordingly set the value of the parameter DFLT_LIFE_CYCLE in the table RSE_CONFIG (APPL_CODE= RSE) from the Manage System Configurations screen. All the items that are *not* supposed to be forecasted by the primary forecast method must be stored in the table RSE_FCST_LIFE_CYCLE_CLSF (populated through an interface). If the customer wants to forecast separately on the items stored in RSE_FCST_LIFE_CYCLE_CLSF and the items that are not, then both Sales & Promo and Life Cycle forecast method run types must be created. The run types will either use the items present in the table RSE_FCST_LIFE_CYCLE_CLSF or use the rest of the items depending on the Forecast Method for the run type and the DFLT_LIFE_CYCLE parameter. You must create a run type for each forecast intersection combination that is required.

Once you have finished setting up the run types, click **Start Data Aggregation** in the Setup train stop. Select all the run types that were created and click **Submit**. When the aggregation is complete, the aggregation status will change to **Complete**. At this point, ad-hoc test runs and batch runs can be created and executed to generate a forecast.

To create an ad-hoc run, go to the **Test** train stop. First, click a run type in the top table and then click the + icon in the bottom table to create a run. In the Create Run pop-up, you can change the configurations parameters related to estimation process, base demand, and forecast process in their respective tabs. After modifying and reviewing the configuration parameters, click **Submit** to start the run. Upon submit, a validation process runs to validate the value of configuration parameters. If there is any error, correct the corresponding parameter and submit again. Once the run is complete, the status will change to Forecast Generation Complete.

Once you are done with testing, you must modify and review the configurations of the run type, activate the run type, enable auto-approve, and map the run type to the downstream application (for example, IPO-DF). In the Manage train stop, select a row, click **Edit Configurations Parameters**, and edit the estimation and forecast parameters as necessary. Once you are done, go to Review tab and click **Validate** and then close tab.

Note

If the run type is active, you will only be able to view the parameters. In order to edit the parameters, the run type must be inactive.

To activate the run type and enable the auto-approve, select a run type in the table and click the corresponding buttons above the table. Lastly, to map the run type to an application (for example, IPO-DF), go to Map train stop and click the + icon to create a new mapping.

By default, the approved forecast source for a run type is AIF. But, when a run type is mapped to IPO-DF, irrespective of what other applications it is mapped to, the approved forecast source changes to RDX. The approved forecast source changes back to AIF when the run type mapping to IPO-DF is deleted. So, the approved forecast source is unique to a run type and changes based on the mapping situation. When a run type is supposed to be mapped to both IPO-DF and IPO-IO, then the "Spread Forecast to Day level" flag should be turned off for the

run type and the forecast will be spread to day level once the approved forecast from IPO-DF is imported back to AIF via RDX. Also, note that when a run type is mapped to RI/RMS, the approved forecast from that run type (either directly from AIF or via IPO-DF depending on whether a mapping also exists to IPO-DF) will be pushed to RI for BI reporting and/or RMS for merchandising. Another nuance is that when a Life Cycle method run type is mapped to LPO-Promotion/Markdown, this run type will treat the value of the DFLT_LIFE_CYCLE parameter as SLC. This enables an environment to forecast on all items using both Sales & Promo and Life Cycle methods, given that the Life Cycle method run type is mapped to LPO-Promotion/Markdown, the RSE_FCST_LIFE_CYCLE_CLSF table is empty, and the DFLT_LIFE_CYCLE parameter is set to LLC.

Batch and Ad-Hoc Jobs for IPO and LPO Forecasting

The configuration and main data load jobs listed in [Table 14-3](#) are used for loading foundation data (product hierarchy, location hierarchy, calendar, product attributes, and so on).

The Batch and Ad-Hoc jobs listed in the following table are used for preparing weekly data and running weekly batches for IPO and LPO.

Table 14-7 Batch and Ad Hoc Jobs for IPO and LPO Forecasting

JobName	Description	RmsBatch
RSE_PROMO_HIER_ETL_STAR T_JOB	Start job for loading promotion hierarchy data	rse_process_state_update.ksh
RSE_PROMO_SRC_XREF_LOA D_JOB	Job for loading promotion hierarchy data	rse_promo_src_xref_load.ksh
RSE_PROMO_HIER_LOAD_JO B	Job for loading promotion hierarchy data	rse_promo_hier_load.ksh
RSE_PROMO_HIER_DH_LOAD _JOB	Job for loading promotion hierarchy data	rse_promo_hier_dh_load.ksh
RSE_PROMO_HIER_ETL_END_ JOB	End job for loading promotion hierarchy data	rse_process_state_update.ksh
PMO_ACTIVITY_LOAD_PL_LLC _START_JOB	Start job for loading promotion lift data	rse_process_state_update.ksh
PMO_ACTIVITY_LOAD_PL_LLC _SETUP_JOB	Setup job for loading promotion lift data	pmo_activity_load_llc_pl_setup.k sh
PMO_ACTIVITY_LOAD_PL_LLC _PROCESS_JOB	Process job for loading promotion lift data	pmo_activity_load_llc_pl_process .ksh
PMO_ACTIVITY_LOAD_PL_LLC _END_JOB	End job for loading promotion lift data	rse_process_state_update.ksh
PMO_ACTIVITY_LOAD_PL_LLC _SETUP_ADHOC_JOB	Ad-hoc setup job for loading promotion lift data	pmo_activity_load_llc_pl_setup.k sh
PMO_ACTIVITY_LOAD_PL_LLC _PROCESS_ADHOC_JOB	Ad-hoc process job for loading promotion lift data	pmo_activity_load_llc_pl_process .ksh
PMO_ACTIVITY_LOAD_OFFER S_INITIAL_ADHOC_JOB	Initial one-time job for loading promotion offers data	pmo_activity_load_offers_initial.k sh
PMO_ACTIVITY_LOAD_PL_LLC _SETUP_JOB	Setup job for loading promotion lift data.	pmo_activity_load_llc_pl_setup.k sh
PMO_ACTIVITY_LOAD_PL_LLC _PROCESS_JOB	Process job for loading promotion lift data	pmo_activity_load_llc_pl_process .ksh
PMO_ACTIVITY_LOAD_OFFER S_START_JOB	Start job for loading promotion offers data	rse_process_state_update.ksh

Table 14-7 (Cont.) Batch and Ad Hoc Jobs for IPO and LPO Forecasting

JobName	Description	RmsBatch
PMO_ACTIVITY_LOAD_OFFER S_LOAD_JOB	Job for loading promotion offers data	pmo_activity_load_offers_weekly.ksh
PMO_ACTIVITY_LOAD_OFFER S_END_JOB	End job for loading promotion offers data	rse_process_state_update.ksh
PMO_ACTIVITY_LOAD_OFFER S_LOAD_ADHOC_JOB	Ad-hoc job for loading promotion offers data	pmo_activity_load_offers_weekly.ksh
RSE_FLEX_GROUP_LOAD_STA RT_JOB	Loading flex group data through interface	rse_process_state_update.ksh
RSE_FLEX_GROUP_DTL_STG_ JOB	Loading flex group data through interface (populates rse_flex_group_dtl_stg)	rse_flex_group_dtl_stg.ksh
RSE_FLEX_GROUP_DTL_COP Y_JOB	Loading flex group data through interface	
RSE_FLEX_GROUP_DTL_STG_ CNE_JOB	Loading flex group data through interface	
RSE_FLEX_GROUP_DTL_LOAD _JOB	Loading flex group data through interface (populates rse_flex_group and rse_flex_group_dtl based on rse_flex_group_set and rse_flex_group_dtl_stg)	rse_flex_group_dtl_load.ksh
RSE_FLEX_GROUP_LOAD_EN D_JOB	Loading flex group data through interface	rse_process_state_update.ksh
RSE_FCST_SPREAD_PROFILE _LOAD_START_JOB	Loading spread profile data through interface	rse_process_state_update.ksh
RSE_FCST_SPREAD_PROFILE _STG_JOB	Loading spread profile data through interface (populates rse_fcst_spread_profile_stg)	rse_fcst_spread_profile_stg.ksh
RSE_FCST_SPREAD_PROFILE _COPY_JOB	Loading spread profile data through interface	
RSE_FCST_SPREAD_PROFILE _STG_CNE_JOB	Loading spread profile data through interface	
RSE_FCST_SPREAD_PROFILE _LOAD_JOB	Loading spread profile data through interface (populates rse_fcst_spread_profile based on rse_fcst_spread_profile_stg)	rse_fcst_spread_profile_load.ksh
RSE_FCST_SPREAD_PROFILE _LOAD_END_JOB	Loading spread profile data through interface	rse_process_state_update.ksh
RSE_FCST_LIFE_CYCLE_CLSF _LOAD_START_JOB	Loading exception life cycle items through interface	rse_process_state_update.ksh
RSE_FCST_LIFE_CYCLE_CLSF _STG_JOB	Loading exception life cycle items through interface (populates rse_fcst_life_cycle_clsf_stg)	rse_fcst_life_cycle_clsf_stg.ksh
RSE_FCST_LIFE_CYCLE_CLSF _COPY_JOB	Loading exception life cycle items through interface	
RSE_FCST_LIFE_CYCLE_CLSF _STG_CNE_JOB	Loading exception life cycle items through interface	

Table 14-7 (Cont.) Batch and Ad Hoc Jobs for IPO and LPO Forecasting

JobName	Description	RmsBatch
RSE_FCST_LIFE_CYCLE_CLSF_LOAD_JOB	Loading exception life cycle items through interface (populates rse_fcst_life_cycle_clsf based on rse_fcst_life_cycle_clsf_stg)	rse_fcst_life_cycle_clsf_load.ksh
RSE_FCST_LIFE_CYCLE_CLSF_LOAD_END_JOB	Loading exception life cycle items through interface	rse_process_state_update.ksh
RSE_INV_LOAD_START_JOB	Inventory Load Start Job	rse_process_state_update.ksh
RSE_INV_PR_LC_WK_STG_JOB	Inventory price location week file stage (populates rse_inv_pr_lc_wk_a_stg)	rse_inv_pr_lc_wk_stg.ksh
RSE_INV_PR_LC_WK_COPY_JOB	Loading price location week file	
RSE_INV_PR_LC_WK_STG_CNE_JOB	Loading price location week file	
RSE_INV_PR_LC_WK_LOAD_JOB	Inventory price location week file load (populates rse_inv_pr_lc_wk_a and rse_inv_pr_lc_hist based on rse_inv_pr_lc_wk_a_stg)	rse_inv_pr_lc_wk_load.ksh
RSE_INV_PR_LC_WK_SETUP_JOB	Inventory price location week AIF Data load setup	rse_inv_pr_lc_wk_setup.ksh
RSE_INV_PR_LC_WK_PROCESS_JOB	Inventory price location week AIF Data load process (populates rse_inv_pr_lc_wk_a and rse_inv_pr_lc_hist based on w_rtl_inv_it_lc_wk_a and w_rtl_invu_it_lc_wk_a)	rse_inv_pr_lc_wk_process.ksh
RSE_INV_LOAD_END_JOB	Inventory Load End Job	rse_process_state_update.ksh
RSE_EXIT_DATE_LOAD_START_JOB	Start job for exit dates	rse_process_state_update.ksh
RSE_EXIT_DATE_STG_JOB	Staging job for exit dates (populates rse_exit_date_stg)	rse_exit_date_stg.ksh
RSE_EXIT_DATE_COPY_JOB	CNE copy job for exit dates	
RSE_EXIT_DATE_STG_CNE_JOB	CNE staging job for exit dates	
RSE_EXIT_DATE_LOAD_JOB	Load job for exit dates (populates rse_exit_date based on rse_exit_date_stg)	rse_exit_date_load.ksh
RSE_EXIT_DATE_LOAD_END_JOB	End job for exit dates	rse_process_state_update.ksh
PMO_RUN_EXEC_START_JOB	Start job for PMO run execution	rse_process_state_update.ksh
*PMO_CREATE_BATCH_RUN_JOB	Create batch run for PMO execution	rse_create_pmo_batch_run_proc.ksh
*PMO_RUN_EXEC_SETUP_JOB	Setup job for PMO run execution	pmo_run_exec_setup.ksh
*PMO_RUN_EXEC_PROCESS_JOB	Process job for PMO run execution	pmo_run_exec_process.ksh
PMO_RUN_EXEC_END_JOB	End job for PMO run execution	rse_process_state_update.ksh
PMO_RUN_EXEC_ADHOC_JOB	Ad-hoc job for PMO run execution	pmo_run_execution.ksh

Table 14-7 (Cont.) Batch and Ad Hoc Jobs for IPO and LPO Forecasting

JobName	Description	RmsBatch
RSE_FCST_BATCH_RUN_START_JOB	Start job for forecast batch run	rse_process_state_update.ksh
RSE_CREATE_FCST_BATCH_RUN_JOB	Create forecast batch run	rse_create_fcst_batch_run_proc.ksh
RSE_FCST_BATCH_PROCESS_JOB	Execute Base Demand and Demand Forecast for forecast run	pmo_new_item_fcst.ksh
RSE_FCST_BATCH_RUN_END_JOB	End job for creating forecast batch run	rse_process_state_update.ksh
RSE_CREATE_FCST_BATCH_RUN_ADHOC_JOB	Adhoc job to create forecast batch run	rse_create_fcst_batch_run_proc.ksh
RSE_FCST_BATCH_PROCESS_ADHOC_JOB	Adhoc job to execute Base Demand and Demand Forecast for forecast run	pmo_new_item_fcst.ksh
*RSE_CALC_SZ_PROF_SC_SKU_JOB	Populates the RSE_SIZE_PROFILE table with size profiles at style-color to sku level	rse_calc_sz_prof_sc_sku.ksh
RSE_CALC_SZ_PROF_SC_SKU_ADHOC_JOB	Adhoc job that populates the RSE_SIZE_PROFILE table with size profiles at style-color to sku level	rse_calc_sz_prof_sc_sku.ksh

The forecast values generated by runs that are approved, associated with active run types and mapped to IPO-DF are exported to RDX schema. The following table lists the jobs to export the main forecast data.

Note

Jobs marked with an asterisk (*) symbol are scheduled to run Quarterly. For example, a quarterly job will run on the first Sunday of each quarter, if Sunday is the day scheduled in the POM UI. If the retailer wants to run the jobs weekly, they have to remove the entries from the RSE_BATCH_FREQ_RESTRICTION table using the Manage System Configuration in the AIF UI.

Table 14-8 Jobs for Exporting Forecast Outputs from AIF to RDX

JobName	Description	RMSBatch
RSE_RDF_FCST_EXPORT_JOB	Export IPO-DF forecast	rse_rdf_export.ksh
RSE_RDF_FCST_EXPORT_ADHOC_JOB	Ad hoc job to export IPO-DF forecast	rse_rdf_export.ksh

The following table lists the additional jobs for exporting different data such as promotions, and configuration parameters for run types.

Table 14-9 Additional Jobs for Exporting Data from AIF to RDX

JobName	Description	RmsBatch
RSE_HIST_PROMO_OFFER_SALES_EXPORT_ADHOC_JOB	Initial one-time job for exporting promotion offer sales figures	rse_rdf_offer_sales_hist_exp.ksh
RSE_PROMO_OFFER_EXPORT_START_JOB	Start job for exporting promotion offer	rse_process_state_update.ksh
RSE_PROMO_OFFER_EXPORT_JOB	Export promotion offers	rse_rdf_offers_hier_exp.ksh
RSE_PROMO_OFFER_SALES_EXPORT_JOB	Export promotion offer sales figures	rse_rdf_offer_sales_exp.ksh
RSE_PROMO_OFFER_EXPORT_END_JOB	End job for exporting promotion offer	rse_process_state_update.ksh
RSE_PROMO_OFFER_EXPORT_ADHOC_JOB	Ad hoc job to export promotion offers	rse_rdf_offers_hier_exp.ksh
RSE_PROMO_OFFER_SALES_EXPORT_ADHOC_JOB	Ad hoc job to export promotion offer sales figures	rse_rdf_offer_sales_exp.ksh
RSE_FCST_RUN_TYPE_CONF_EXPORT_START_JOB	Start job for exporting runtype config	rse_process_state_update.ksh
RSE_FCST_RUN_TYPE_CONF_EXPORT_SETUP_JOB	Setup job for exporting runtype config	rse_fcst_run_type_conf_exp_setup.ksh
RSE_FCST_RUN_TYPE_CONF_EXPORT_PROCESS_JOB	Process job for exporting runtype config	rse_fcst_run_type_conf_exp_process.ksh
RSE_FCST_RUN_TYPE_CONF_EXPORT_END_JOB	End job for exporting runtype config	rse_process_state_update.ksh
RSE_FCST_RUN_TYPE_CONF_EXPORT_SETUP_ADHOC_JOB	Ad-hoc setup job for exporting runtype config	rse_fcst_run_type_conf_exp_setup.ksh
RSE_FCST_RUN_TYPE_CONF_EXPORT_PROCESS_ADHOC_JOB	Ad-hoc process job for exporting runtype config	rse_fcst_run_type_conf_exp_process.ksh

The jobs for importing forecast parameters from RDX schema are listed in the following table.

Table 14-10 Jobs for Importing Forecast Parameters from RDX to AIF

JobName	Description	RMSBatch
RSE_RDX_FCST_PARAM_START_JOB	Start job for importing forecast parameters	rse_process_state_update.ksh
RSE_RDX_FCST_PARAM_SETUP_JOB	Setup job for importing forecast parameters	rse_rdx_fcst_param_setup.ksh
RSE_RDX_FCST_PARAM_PROCESS_JOB	Process job for importing forecast parameters	rse_rdx_fcst_param_process.ksh
RSE_RDX_FCST_PARAM_END_JOB	End job for importing forecast parameters	rse_process_state_update.ksh
RSE_RDX_FCST_PARAM_SETUP_ADHOC_JOB	Ad-hoc setup job for importing forecast parameters	rse_rdx_fcst_param_setup.ksh
RSE_RDX_FCST_PARAM_PROCESS_ADHOC_JOB	Ad-hoc process job for importing forecast parameters	rse_rdx_fcst_param_process.ksh

Table 14-10 (Cont.) Jobs for Importing Forecast Parameters from RDX to AIF

JobName	Description	RMSBatch
RSE_FCST_RDX_NEW_ITEM_ENABLE_START_JOB	Start job for importing enablement flags for new item forecast	rse_process_state_update.ksh
RSE_FCST_RDX_NEW_ITEM_ENABLE_SETUP_JOB	Setup job for importing enablement flags for new item forecast	rse_fcst_rdx_new_item_enable_setup.ksh
RSE_FCST_RDX_NEW_ITEM_ENABLE_PROCESS_JOB	Process job for importing enablement flags for new item forecast	rse_fcst_rdx_new_item_enable_process.ksh
RSE_FCST_RDX_NEW_ITEM_ENABLE_END_JOB	End job for importing enablement flags for new item forecast	rse_process_state_update.ksh
RSE_FCST_RDX_NEW_ITEM_ENABLE_SETUP_ADHOC_JOB	Ad-hoc setup job for importing enablement flags for new item forecast	rse_fcst_rdx_new_item_enable_setup.ksh
RSE_FCST_RDX_NEW_ITEM_ENABLE_PROCESS_ADHOC_JOB	Ad-hoc process job for importing enablement flags for new item forecast	rse_fcst_rdx_new_item_enable_process.ksh
PMO_EVENT_IND_RDF_START_JOB	Start job for importing event indicators for preprocessing	rse_process_state_update.ksh
PMO_EVENT_IND_RDF_SETUP_JOB	Setup job for importing event indicators for preprocessing	pmo_event_ind_rdf_setup.ksh
PMO_EVENT_IND_RDF_PROCESS_JOB	Process job for importing event indicators for preprocessing	pmo_event_ind_rdf_process.ksh
PMO_EVENT_IND_RDF_END_JOB	End job for importing event indicators for preprocessing	rse_process_state_update.ksh
PMO_EVENT_IND_RDF_SETUP_ADHOC_JOB	Ad-hoc setup job for importing event indicators for preprocessing	pmo_event_ind_rdf_setup.ksh
PMO_EVENT_IND_RDF_PROCESS_ADHOC_JOB	Ad-hoc process job for importing event indicators for preprocessing	pmo_event_ind_rdf_process.ksh
RSE_LIKE_RDX_RSE_START_JOB	Start job for importing new item parameters	rse_process_state_update.ksh
RSE_LIKE_RDX_RSE_SETUP_JOB	Setup job for importing new item parameters	rse_like_rdx_rse_setup.ksh
RSE_LIKE_RDX_RSE_PROCESS_JOB	Process job for importing new item parameters	rse_like_rdx_rse_process.ksh
RSE_LIKE_RDX_RSE_END_JOB	End job for importing new item parameters	rse_process_state_update.ksh
RSE_LIKE_RDX_RSE_SETUP_ADHOC_JOB	Ad-hoc setup job for importing new item parameters	rse_like_rdx_rse_setup.ksh
RSE_LIKE_RDX_RSE_PROCESS_ADHOC_JOB	Ad-hoc process job for importing new item parameters	rse_like_rdx_rse_process.ksh

The approved forecast is exported out of RDX into AIF, which then goes to RMS via flat files. The jobs are listed in the following table. Note that the approved forecast source for a run type depends on whether it is mapped to the IPO-DF application. If this mapping exists, then the approved forecast source for the run type is RDX, otherwise it is AIF.

Table 14-11 Approved Forecast Export Jobs

JobName	Description	RMSBatch
RSE_RDX_APPD_FCST_START_JOB	Start job for exporting approved forecast data	rse_process_state_update.ksh
RSE_RDX_APPD_FCST_SETUP_JOB	Setup job for exporting approved forecast data	rse_rdx_appd_fcst_setup.ksh
RSE_RDX_APPD_FCST_PROCESS_JOB	Process job for exporting approved forecast data	rse_rdx_appd_fcst_process.ksh
RSE_SPREAD_APPD_RDF_FCST_TO_DAY_JOB	Job to spread down approved forecast to day level	rse_spread_rdf_fcst_to_day.ksh
RSE_RDX_APPD_FCST_END_JOB	End job for exporting approved forecast data	rse_process_state_update.ksh
RSE_RDF_APPR_FCST_EXPORT_JOB	Export approved weekly forecast to a flat file	rse_rdf_appr_fcst_export.ksh
RSE_RDF_APPR_FCST_DAY_EXPORT_JOB	Export approved daily forecast to a flat file	rse_rdf_appr_fcst_day_export.ksh
RSE_RDX_APPD_FCST_SETUP_ADHOC_JOB	Ad-hoc setup job for exporting approved forecast data	rse_rdx_appd_fcst_setup.ksh
RSE_RDX_APPD_FCST_PROCESS_ADHOC_JOB	Ad-hoc process job for exporting approved forecast data	rse_rdx_appd_fcst_process.ksh
RSE_SPREAD_APPD_RDF_FCST_TO_DAY_ADHOC_JOB	Ad hoc job to spread down approved forecast to day level	rse_spread_rdf_fcst_to_day.ksh
RSE_RDF_APPR_FCST_EXPORT_ADHOC_JOB	Ad hoc job to export approved weekly forecast to a flat file	rse_rdf_appr_fcst_export.ksh
RSE_RDF_APPR_FCST_DAY_EXPORT_ADHOC_JOB	Ad-hoc job to export approved daily forecast to a flat file	rse_rdf_appr_fcst_day_export.ksh

Loading and Calculating Event Flags in AIF

The Out-of-Stock and Outlier indicators are important for the estimation stage of forecasting. These two flags are stored in the AIF table PMO_EVENT_INDICATOR that is used by the Sales & Promo forecast method.

Loading Out-of-Stock and Outlier Indicators Through the RAP Interface

The customer can load Out-of-Stock and Outlier indicators at any forecast level using the RAP interface (staging table: W_RTL_INVOOS_IT_LC_WK_FS; target table: W_RTL_INVOOS_IT_LC_WK_F). The indicators will flow into AIF that will be then used in forecasting.

Loading Out-of-Stock and Outlier Indicators Through the RDX Interface

This is implemented via a package RAP_INTF_SUPPORT_UTIL which is also granted to IW, and through the use of granting DML permissions on all the interface tables that reside in RDX01. In order to publish data for an interface, the implementer must first know some things about the interface being published. For this specific use case, the interface name is RDF_FCST_PARM_CAL_EXP.

```
SELECT * FROM RAP_INTF_CFG WHERE INTF_NAME = 'RDF_FCST_PARM_CAL_EXP';
```


Of the output from the above query, the PUBLISH_APP_CODE is important to pay attention to (for this example, the value is PDS). Also, the value in CUSTOMER_PUBLISHABLE_FLG is important. In order to publish data to these interfaces, it is necessary to have this value be a Y for any interface that IW is going to be populating. For customer implementations, this would require requesting AMS to update the value to a Y, like this:

```
UPDATE RAP_INTF_CFG SET CUSTOMER_PUBLISHABLE_FLG = 'Y' WHERE INTF_NAME
= 'RDF_FCST_PARM_CAL_EXP'; COMMIT;
```

Once the above has been done, then the implementer/customer can write code in the IW that does something like this:

```
DECLARE
v_run_id      NUMBER;
v_partition_name  VARCHAR2(30);
BEGIN
RAP_INTF_SUPPORT_UTIL.PREP_INTF_RUN('PDS', 'RDF_FCST_PARM_CAL_EXP', v_run_id,
v_partition_name);
--Note, the PDS value is what was obtained from the above RAP_INTF_CFG query.
EXECUTE IMMEDIATE 'INSERT /*+ append */ INTO RDF_FCST_PARM_CAL_EXP partition
( ||v_partition_name|| ) .... ';
--NOTE, the .... is to be replaced with the actual list of columns and an
appropriate SELECT statement to provide the values.
--Of importance, it should be noted that the RUN_ID column must be populated
with the value that was returned into the v_run_id variable above.
--After the data has been populated into the above table, it can be made
available for retrieval by consuming application modules by the following:
RAP_INTF_SUPPORT_UTIL.PUBLISH_DATA('PDS', v_run_id, 'RDF_FCST_PARM_CAL_EXP',
'Ready');
END;
```

Once the above steps have been completed, the data will now be ready to be consumed by the application modules which use this interface.

Note

If an attempt is made to call any of the routines inside RAP_INTF_SUPPORT_UTIL for an interface that was not updated so that CUSTOMER_PUBLISHABLE_FLG = Y, then an error will be provided indicating that "No Interfaces for [<values they provided to the procedure>] are allowed/available to be published from custom code.

Calculating Outlier Indicators in AIF

Outlier indicators can also be calculated directly in AIF at any forecast level. A historical sale for a particular Product/Location/Week is flagged as an outlier if it is more than the last 52-week average rate-of-sales for the Product/Location times a threshold. The threshold value is provided using the parameter PP_LLC_OUTLIER_THRESHOLD in the Preprocessing tab while setting up a Sales & Promo forecast run in Manage Forecast Configurations screen. This feature can be turned on/off using the parameter PMO_OUTLIER_CALCULATION_FLG in the table RSE_CONFIG (APPL_CODE = PMO) from the Manage System Configurations screen. It is expected that an environment will either load outlier indicators through the RAP/RDX interface or calculate in AIF.

Loading and Calculating Like Products and Like Locations in AIF

The like products and the like locations are important to calculate the base demand for new products and new locations, respectively. This data is stored in the AIF table RSE_FCST_NEW_ITEM_PARAM that is used by the Sales & Promo and Life Cycle forecast methods. The run type parameter RSE_FCST_NEW_ITEM_STRATEGY determines the strategy used to generate the forecast for new items. This parameter can be configured in the Manage (TS3) screen. The following four options are available for the RSE_FCST_NEW_ITEM_STRATEGY parameter.

1. Loading Like Products and Like Locations Through the RAP Interface

The customer can load like products and like locations at any forecast level using the RAP interface (staging table: W_RTL_LIKE_IT_LC_DS; target table: W_RTL_LIKE_IT_LC_D). When the parameter RSE_FCST_NEW_ITEM_STRATEGY in the table RSE_CONFIG (APPL_CODE = RSE) from the Manage System Configurations screen is set as RI, this data will flow into AIF which will be then used in forecasting.

2. Loading Like Products and Like Locations Through the RDX Interface

This is implemented via a package RAP_INTF_SUPPORT_UTIL and the interface RDF_NIT_PARM_EXP. (See [Loading Out-of-Stock and Outlier Indicators Through the RDX Interface](#) for more details.) When the parameter RSE_FCST_NEW_ITEM_STRATEGY in the table RSE_CONFIG (APPL_CODE = RSE) from the Manage System Configurations screen is set as RDX, this data will flow into AIF which will be then used in forecasting.

3. Calculating Like Products in AIF

Like products can be calculated in AIF only at the SKU level based on attribute and image similarities. This process starts by finding the new SKUs automatically based on the parameter RSE_FCST_NEW_ITEM_AGE_THRESH in the table RSE_CONFIG (APPL_CODE = RSE) from the Manage System Configurations screen. In addition to setting the parameter RSE_FCST_NEW_ITEM_STRATEGY in the table RSE_CONFIG (APPL_CODE = RSE) from the Manage System Configurations screen to RSP, set up the Similarity tab appropriately while creating a new forecast run in the Manage Forecast Configurations screen for this data to be used in forecasting. The following table lists some jobs to calculate the product/location range data and the baseline sales. This is required to prepare the data for calculating attribute weights and attribute similarities.

Table 14-12 Jobs to Calculate the Product/Location Range Data and the Baseline Sales

JobName	Description	RMSBatch
RSE_PROD_LOC_RANGE_SETUP_JOB	Job to set up a queue for RSE_PROD_LOC_RANGE service that populates the RSE_PROD_LOC_STATUS table with product/location range data using data in the W_RTL_IT_LC_D/ RSE_SLS_PR_LC_WK tables	rse_prod_loc_range_setup.ksh

Table 14-12 (Cont.) Jobs to Calculate the Product/Location Range Data and the Baseline Sales

JobName	Description	RMSBatch
RSE_PROD_LOC_RANGE_PRO CESS_JOB	Job to process a queue for RSE_PROD_LOC_RANGE service that populates the RSE_PROD_LOC_STATUS table with product/location range data using data in the W_RTL_IT_LC_D/ RSE_SLS_PR_LC_WK tables	rse_prod_loc_range_process.ksh
RSE_PROD_LOC_RANGE_SET UP_ADHOC_JOB	Ad hoc job to set up a queue for RSE_PROD_LOC_RANGE service that populates the RSE_PROD_LOC_STATUS table with product/location range data using data in the W_RTL_IT_LC_D/ RSE_SLS_PR_LC_WK tables	rse_prod_loc_range_setup.ksh
RSE_PROD_LOC_RANGE_PRO CESS_ADHOC_JOB	Ad hoc job to process a queue for RSE_PROD_LOC_RANGE service that populates the RSE_PROD_LOC_STATUS table with product/location range data using data in the W_RTL_IT_LC_D/ RSE_SLS_PR_LC_WK tables	rse_prod_loc_range_process.ksh
RSE_BASELINE_SETUP_JOB	Job to set up a queue for RSE_BASELINE service that executes baseline sales calculation using data in the RSE_SLS_PR_LC_CS_WK/ RSE_PROD_LOC_STATUS tables and populates the RSE_BL_SLS_PR_LC_CS_WK table	rse_baseline_setup.ksh
RSE_BASELINE_PROCESS_JO B	Job to process a queue for RSE_BASELINE service that executes baseline sales calculation using data in the RSE_SLS_PR_LC_CS_WK/ RSE_PROD_LOC_STATUS tables and populates the RSE_BL_SLS_PR_LC_CS_WK table	rse_baseline_process.ksh
RSE_BASELINE_SETUP_ADHO C_JOB	Ad hoc job to set up a queue for RSE_BASELINE service that executes baseline sales calculation using data in the RSE_SLS_PR_LC_CS_WK/ RSE_PROD_LOC_STATUS tables and populates the RSE_BL_SLS_PR_LC_CS_WK table	rse_baseline_setup.ksh

Table 14-12 (Cont.) Jobs to Calculate the Product/Location Range Data and the Baseline Sales

JobName	Description	RMSBatch
RSE_BASELINE_PROCESS_AD HOC_JOB	Ad hoc job to process a queue for RSE_BASELINE service that executes baseline sales calculation using data in the RSE_SLS_PR_LC_CS_WK/ RSE_PROD_LOC_STATUS tables and populates the RSE_BL_SLS_PR_LC_CS_WK table	rse_baseline_process.ksh

4. Combination of Loading and Calculating Like Products in AIF

This option enables the user to load the new item strategy using the RI interface or populate it using the New Item/Location Setup screens. It programmatically identifies new items based on LLC_BD_TIME_SERIES_DURATION and generates like items based on attributes. The RSE_FCST_NEW_ITEM_STRATEGY parameter should be set to RI_RSP. Additional parameters related to new item and location can be configured in the Manage (TS3) screen. The New Item/Location Setup screens are editable only if the RSE_FCST_NEW_ITEM_STRATEGY = RI_RSP. The user can set up or override the system-generated like items using the New Item Setup screen.

Item Replacement / Supersession in AIF

This functionality covers the two use cases Substitute Item and Replacement Item:

Substitute Item

Sometimes an item is introduced to fill a gap period when the supplier cannot fulfill the replenishment order.

The sales of the substitute item contributes towards the forecast of the primary item.

Replacement Item

An item is being phased out and a new item which is virtually identical to the existing item is set up as a replacement. The replacement item begins selling alongside the discontinued product and the combined sales contributes towards the forecast of the replacement item.

Technical Details

The Item Supersession mapping information can flow from MFCS or can be loaded by using the RI interface W_RTL_SUB_IT_LC_DS.dat. When loading using the RI interface, the following POM jobs populate the DB tables:

W_RTL_SUB_IT_LC_DS_COPY_JOB populates W_RTL_SUB_IT_LC_DS

W_RTL_SUB_IT_LC_DS_STG_JOB populates W_RTL_SUB_IT_LC_DS

The parameter ITEM_REPL_FLG is used to control this functionality. It can be enabled per run type using the Manage (TS3) screen. Default value is N.

AIF table RSE_FCST_REPL_ITEM holds the Replacement & Substitution (RI) data. The POM job RSE_FCST_REPL_ITEM_SETUP_JOB populates RSE_FCST_REPL_ITEM from W_RTL_SUB_IT_LC_D.

The I REPLACEMENT_ITEM_FLG parameter can be used with the RSE_MASTER adhoc job.

Once the ITEM_REPL_FLG has been enabled per run type and the RSE_FCST_REPL_ITEM table populated, forecast for the primary item will be populated in the regular forecast tables.

Workflow for IPO-DF Implementation

The AIF-IPO-DF workflow can be implemented in two ways.

The main dependency for the first approach is that even before exporting first round of forecast from AIF to RDX, parameters need to be imported from RDX to AIF (the reason for this is IPO-DF prefers to receive forecast for a subset of the product/locations for which forecast is generated in AIF). For the import of parameters to work properly, run types must be mapped to IPO-DF and assigned an external key. Here are the steps for the first approach:

1. Create run types in AIF in Setup train stop screen.
2. Create test runs in AIF in Test train stop screen to see if the setup has been correct.
3. Map run types to IPO-DF and assign external keys in Map train stop screen.
4. Import parameters from RDX to AIF (IPO-DF sends parameters using the same external keys used in Step 3).
5. Run forecasts again in Test train stop screen (this time, the forecasts will use the imported parameters).
6. Approve the estimation and forecast runs in Test train stop screen.
7. Activate the run types in Manage train stop screen.
8. Export forecasts from AIF to RDX.
9. Cycle continues:
 - a. Import parameters from RDX to AIF.
 - b. Run forecasts.
 - c. Approve runs.
 - d. Export forecasts from AIF to RDX.

In the second approach, there is no dependency of importing parameters from RDX to AIF before exporting forecast from AIF to RDX. (This simplifies the workflow and saves time during implementation; however, the batch processes will still have the dependency.) Here are the steps for the second approach:

1. Create run types in AIF in Setup train stop screen.
2. Create test runs in AIF in Test train stop screen to see if the setup has been correct.
3. Approve the estimation and forecast runs in Test train stop screen.
4. Activate the run types in Manage train stop screen.
5. Map run types to IPO-DF and assign external keys in Map train stop screen.
6. Export forecasts from AIF to RDX (this will export all product/locations for which AIF has generated forecast; however, this export will take longer time because it is a huge amount of data; this is the tradeoff between the two implementation approaches).

7. Cycle continues as in Step 9 of the first approach.

The internal (w.r.t. AIF) difference between the two approaches is as follows: When the table in AIF that stores imported parameters from RDX is empty, AIF will export forecast for all product/locations (second approach); otherwise, AIF will export forecast for only a subset of product/locations as imported from RDX (first approach).

Note that once a forecast run has been exported from AIF to RDX, it can't be exported again. For an active run type, the export code finds the latest forecast run that has completed, has been approved, and has not been exported before.

AIF-IPO-DF Workflow Example

Here is a set of sample steps using the first approach.

1. In AIF Manage Forecast Configurations train stop 1 Setup, create Run Type.
 - For forecasts used by IPO-DF, choose Sales & Promo (preferred) or Life Cycle as the Forecast Method.
 - If any run type has an aggregation status of Not Started, click the Start Data Aggregation button for the new run type just created, this process can take a few hours to complete.
2. In AIF Manage Forecast Configurations train stop 2 Test.
 - Once aggregation completes, create a new run under selected run type, choose to run Estimation and Base Demand in this step, you do not have to toggle to Generate Forecast initially.
 - Approve the estimation parameters by clicking on Approve Demand Parameters.
 - Tables to check:
 - Seasonality (from estimation)
 - * For both methods: RSE_SEAS_CURVE_DIM (using pmo run id)
 - * For both methods: RSE_SEAS_CURVE_WK (using pmo run id)
 - Base Demand
 - * For Life Cycle: RSE_BASE_DEMAND (using fcst run id)
 - * For Sales & Promo: RSE_LLC_BD_RES (using fcst run id)
 - Forecast (when Generate Forecast is selected)
 - * For Life Cycle: RSE_DMD_FCST (using fcst run id)
 - * For Sales & Promo: RSE_LLC_FCST_BLINE (using fcst run id)
3. In AIF Manage Forecast Configurations train stop 4 Map.
 - Map the forecast run type to IPO-DF
 - Make sure to assign external application key for IPO-DF, which is usually Weekly Forecast (this name has to be matched to IPO-DF side)
4. In IPO-DF:
 - Perform your initial IPO-DF Workflow Tasks to set up forecast parameters (History Length, Forecast Start Date, Forecast End Date), new items, etc.
 - Run the IPO-DF Pre-Forecast and Export Forecast Parameters Batches using IPO_PRE_BATCH_W_JOB and IPO_PRE_EXP_RDX_W_JOB processes.

- POM job IPO_PRE_BATCH_W_JOB triggers Pre-Forecast Batch task in IPO-DF OAT
- POM job IPO_PRE_EXP_RDX_W_JOB triggers Export Forecast Parameters task in IPO-DF OAT
- Check the following tables after the batches complete.
 - * RDF_FCST_PARM_HDR_EXP
 - * rdf_rdfruntype01:RDF_RUN_TYPE_KEY
 - * RDF_RUN_TYPE_PARM_EXP
 - * rdf_rdfruntype01:RDF_RUN_TYPE_KEY
 - * rdf_runnewitem01:ENABLE_NEW_ITEM
 - * rdf_runfrcst01:ENABLE_FORECAST
 - * RDF_FCST_PARM_EXP
 - * rdf_hislen01:FCST_HIST_LENGTH
 - * rdf_frcststartdt01:FCST_START_DATE
 - * rdf_frcstenddtlw01:FCST_END_DATE
 - * RDF_FCST_PARM_CAL_EXP
 - * rdf_ppsbopdate01:BOP_DATE
 - * rdf_totadjslsovr01:USER_OVERRIDE
 - * rdf_outageind01:PPS_OUTAGE_IND
 - * rdf_outlierind01:PPS_OUTLIER_IND
 - * RDF_NIT_PARM_EXP
 - * PROD_HIER_LEVEL:SKU
 - * LOC_HIER_LEVEL:STOR
 - * rdf_fcpsubm:NIT_SUB_METHOD
 - * rdf_fcpros:NIT_ROS_USER_IN
 - * rdf_nitapplkitm1:NIT_LIKE_ITEM
 - * rdf_nstapplkstr1:NIT_LIKE_STOR
 - * rdf_fcpadj:ADJUSTMENT_FACTOR
- 5. Run jobs in POM to import parameters from RDX to AIF (during the weekly batch this transition from RPASCE to AIF is managed by the dependency between IPO_PRE_EXP_PROCESS and PMO_PRE_ESTIM_CUSTOM_PROCESS, specifically by the jobs IPO_PRE_EXP_END_JOB and PMO_PRE_ESTIM_CUSTOM_START_JOB).
 - RSE_RDX_FCST_PARAM_ADHOC_PROCESS
 - RSE_RDX_FCST_PARAM_SETUP_ADHOC_JOB
 - RSE_RDX_FCST_PARAM_PROCESS_ADHOC_JOB
 - Table to check:
 - * RSE_RDF_FCST_RUN_TYPE_PRM (from RDF_FCST_PARM_HDR_EXP and RDF_FCST_PARM_EXP)
 - RSE_FCST_RDX_NEW_ITEM_ENABLE_ADHOC_PROCESS
 - RSE_FCST_RDX_NEW_ITEM_ENABLE_SETUP_ADHOC_JOB

- RSE_FCST_RDX_NEW_ITEM_ENABLE_PROCESS_ADHOC_JOB
 - Table to check:
 - * RSE_FCST_RDX_NEW_ITEM_ENABLE (from RDF_RUN_TYPE_PARM_EXP)
 - RSE_LIKE_RDX_RSE_ADHOC_PROCESS
 - RSE_LIKE_RDX_RSE_SETUP_ADHOC_JOB
 - RSE_LIKE_RDX_RSE_PROCESS_ADHOC_JOB
 - Table to check:
 - * RSE_FCST_NEW_ITEM_PARAM (from RDF_NIT_PARM_EXP)
 - PMO_EVENT_IND_RDF_ADHOC_PROCESS
 - PMO_EVENT_IND_RDF_SETUP_ADHOC_JOB
 - PMO_EVENT_IND_RDF_PROCESS_ADHOC_JOB
 - Table to check:
 - * PMO_EVENT_INDICATOR (from RDF_FCST_PARM_CAL_EXP)
6. In AIF Manage Forecast Configurations train stop 2 Test.
- Create a new run under the same run type you have been working with, choose to run Base Demand only if there is no need to run Estimation, and toggle to Generate Forecast this time.
 - If estimation has been run, approve the Estimation parameters by clicking on Approve Demand Parameters
 - Approve the base demand and forecast by clicking on Approve Base Demand and Forecast
 - Tables to check:
 - Seasonality (when estimation is run):
 - * For both methods: RSE_SEAS_CURVE_DIM (using pmo run id)
 - * For both methods: RSE_SEAS_CURVE_WK (using pmo run id)
 - Base Demand
 - * For Life Cycle: RSE_BASE_DEMAND (using fcst run id)
 - * For Sales & Promo: RSE_LLC_BD_RES (using fcst run id)
 - Forecast
 - * For Life Cycle: RSE_DMD_FCST (using fcst run id)
 - * For Sales & Promo: RSE_LLC_FCST_BLINE (using fcst run id)
7. In AIF Manage Forecast Configurations train stop 3 Manage.
- View/Edit Configuration Parameters to see/apply parameters from previously approved run (these parameters will be used for batch runs; not needed for forecast runs created from AIF UI)
 - Activate the forecast run type (after this to ensure that the data is synced up between the source table (for example, RSE_SLS_PR_LC_CS_WK when Data Source is Store Sales) and PMO_ACTIVITIES for the appropriate AGGR_LVL_KEY; if not, then execute the relevant jobs in PMO_ACTIVITY_LOAD_ADHOC_PROCESS with the necessary parameters to sync up the data)

- Manage Auto-Approve to enable Auto-Approve for the run type before starting batch cycles (not needed for forecast runs created from AIF UI)
8. Run jobs in POM to export forecasts to RDX from AIF.
 - RSE_FCST_RUN_TYPE_CONF_EXPORT_ADHOC_PROCESS
 - RSE_FCST_RUN_TYPE_CONF_EXPORT_SETUP_ADHOC_JOB
 - RSE_FCST_RUN_TYPE_CONF_EXPORT_PROCESS_ADHOC_JOB
 - RSE_FCST_EXPORT_ADHOC_PROCESS (Note that this process also includes the export of AP and MFP forecasts. For IPO-DF only, run the following export job.)
 - RSE_RDF_FCST_EXPORT_ADHOC_JOB
 9. After the jobs complete, check the following tables in Innovation Workbench to do a quick validation.
 - Table: rdx01.rse_fcst_run_type_config (gets populated by RSE_FCST_RUN_TYPE_CONF_EXPORT_ADHOC_PROCESS; based on an independent export run id)
 - Table: rdx01.rse_fcst_run_hdr_exp (gets populated by RSE_RDF_FCST_EXPORT_ADHOC_JOB; this table associates the export run id to the forecast run id approved in Step 6)
 - Table: rdx01.rse_fcst_run_config (gets populated by RSE_RDF_FCST_EXPORT_ADHOC_JOB; based on the export run id from rdx01.rse_fcst_run_hdr_exp)
 - Table: rdx01.rse_fcst_appr_base_dmd_exp (gets populated by RSE_RDF_FCST_EXPORT_ADHOC_JOB; based on the export run id from rdx01.rse_fcst_run_hdr_exp)
 - Table: rdx01.rse_fcst_demand_dtl_exp (gets populated by RSE_RDF_FCST_EXPORT_ADHOC_JOB; based on the export run id from rdx01.rse_fcst_run_hdr_exp)
 - Table: rdx01.rse_fcst_demand_src_exp (gets populated by RSE_RDF_FCST_EXPORT_ADHOC_JOB; based on the export run id from rdx01.rse_fcst_run_hdr_exp)
 - Table: rdx01.rse_fcst_demand_dtl_cal_exp (gets populated by RSE_RDF_FCST_EXPORT_ADHOC_JOB; based on the export run id from rdx01.rse_fcst_run_hdr_exp)
 10. In IPO-DF:
 - Import the forecasts to IPO-DF using the IPO_POST_DATA_IMP_PROCESS. Run all the jobs like IPO_POST_DATA_IMP_*. IPO_POST_DATA_IMP_RDX_W is the corresponding OAT task within IPO-DF that imports the forecast.
 - RSE_FCST_RUN_HDR_EXP
 - * rdf_rserunid01:RUN_ID
 - RSE_FCST_DEMAND_DTL_CAL_EXP
 - * rdf_sysbaseline01:BASELINE_FCST_QTY
 - * rdf_sysfcst01:DEMAND_FCST_QTY
 - * rdf_syspeak01:EVENT_PEAK_QTY
 - * rdf_eventfut01:EVENT_CLND
 - * rdf_prctdiscfut01:PRICE_DISC_PERCENT

- RSE_FCST_DEMAND_DTL_EXP
 - * rdf_basedemand01:BASE_DEMAND_QTY
 - * rdf_priceelas01:PRICE_ELASTICITY
- RSE_FCST_APPR_BASE_DMD_EXP
 - * rdf_appchosenlevel01:APPR_BASE_DEMAND_QTY
- RSE_FCST_DEMAND_SRC_EXP
 - * rdf_outageind01:STOCKOUT_IND
 - * rdf_outageadj01:LOST_SLS_QTY
 - * rdf_outlierind01:OUTLIER_IND
 - * rdf_outlieradj01:OUTLIER_SLS_QTY
 - * rdf_depromoadj01:PROMO_SLS_QTY
 - * rdf_deprice01:CLR_SLS_QTY
 - * rdf_totadjsls01:PREPROCESSED_SLS_QTY
- Once the above jobs complete successfully, validate that the above interfaces have been imported successfully into IPO-DF. Check the RAP_INTF_CFG table to get the INTF_CFG_ID. Then check the RAP_INTF_RUN_STATUS table to make sure that the interface is marked at SUCCESS for the PDS import for the latest run_id.
- Run the IPO_POST_BATCH_PROCESS using POM to execute the Post Forecast Batch in IPO-DF. IPO_POST_BATCH_W_JOB is the corresponding job.
- Build Forecast Review workspace to review the forecasts, review, update and approve forecasts
- Export approved forecasts by running IPO_POST_EXP_RDX POM job
 - Export Approved Forecast OAT Task
 - * RDF_APPR_FCST_HDR_EXP
 - * rdf_apprserunid01:RSE_RUN_ID
 - * RDF_APPR_FCST_CAL_EXP
 - * rdf_bopdate01:BOP_DATE
 - * rdf_appbaseline01:APPR_BASELINE_FCST
 - * rdf_appfrcst01:APPR_DEMAND_FCST
 - * rdf_appcumint01:APPR_CUMINT
- 11. To push the approved forecast from RDX to RMS/object store (as a flat file) via AIF(during the weekly/daily batch this transition from RPASCE to AIF is managed by the dependency between IPO_POST_EXP_PROCESS and RSE_RDX_APPD_FCST_PROCESS, specifically by the jobs IPO_POST_EXP_END_JOB and RSE_RDX_APPD_FCST_START_JOB).
 - Run RSE_RDX_APPD_FCST_ADHOC_PROCESS in POM to import approved forecast from RDX to AIF:
 - RSE_RDX_APPD_FCST_SETUP_ADHOC_JOB
 - RSE_RDX_APPD_FCST_PROCESS_ADHOC_JOB
 - RSE_SPREAD_APPD_RDF_FCST_TO_DAY_ADHOC_JOB (this job spreads down the approved weekly forecast to day level and the “Forecast day level

spread horizon in weeks" parameter in the Scope tab while creating a forecast run controls how many weeks of forecast from the approved forecast start week will be spread down; in addition, for this job to work the run type also needs to be mapped to the IPO-IO application with the run type having the "Spread Forecast to Day level" toggle button turned OFF when the run type is being created)

– Table to check:

- * RSE_RDF_APPR_FCST (from RDF_APPR_FCST_HDR_EXP and RDF_APPR_FCST_CAL_EXP)
 - Run RSE_RDF_APPR_FCST_EXPORT_ADHOC_PROCESS (RSE_RDF_APPR_FCST_EXPORT_ADHOC_JOB) in POM to export the data from RSE_RDF_APPR_FCST_VW (week-level forecast) to RMS/object store as a flat file (the APPR_FCST_EXPORT_NUM_WK parameter value in RSE_CONFIG table (APPL_CODE = RSE) available in the Manage System Configurations screen controls how many weeks of forecast from the forecast start week will be exported).
 - Run RSE_RDF_APPR_FCST_DAY_EXPORT_ADHOC_PROCESS (RSE_RDF_APPR_FCST_DAY_EXPORT_ADHOC_JOB) in POM to export the data from RSE_RDF_APPR_FCST_DAY_VW (day-level forecast) to RMS/object store as a flat file, if day-level forecast is generated (the APPR_FCST_DAY_EXPORT_NUM_DY parameter value in RSE_CONFIG table (APPL_CODE = RSE) available in the Manage System Configurations screen controls how many days of forecast from the forecast start day will be exported).
12. Run the AIF DATA standalone processes LOAD_PLANFC1_DATA_ADHOC and LOAD_PLANFC2_DATA_ADHOC to import the forecasts into RI for reporting. The PLANFC1 jobs will obtain the AIF base demand when IPO-DF is used, and otherwise do nothing otherwise. The PLANFC2 jobs source from RSE_RDF_APPR_FCST_VW, which contains only the weekly approved forecast export.

Note

For the approved forecast to flow from RSE_RDF_APPR_FCST to RSE_RDF_APPR_FCST_VW and RSE_RDF_APPR_FCST_DAY_VW, the run types also need to be mapped to the RI/RMS application in AIF Manage Forecast Configurations train stop 4 Map.

Note

Refer to Table C-1 List of Jobs in IPOCS- Demand Forecasting / IPOCS-Lifecycle Allocation and Replenishment Schedule in the *Oracle Retail Inventory Planning Optimization Cloud Service-Demand Forecasting / Inventory Planning Optimization Cloud Service-Lifecycle Allocation and Replenishment Implementation Guide*.

Using the Add Multiple Run Types Feature

An "Add Multiple Run Types" feature is available in the Setup train stop within the Manage Forecast Configurations screen. To add/edit/delete the rows visible within the "Add Multiple Run Types" table, please edit the two tables, RSE_CUSTOM_PROCESS_HDR and RSE_CUSTOM_PROCESS_DTL, available in the Manage System Configurations screen.

Building an Alternate Hierarchy in AIF Applications

AIF applications have the ability to build alternate location and alternate product hierarchies. Alternate hierarchy information should be available in the AIF data warehouse tables (also used by Retail Insights). RSE_ALT_HIER_TYPE_STG and RSE_ALT_HIER_LEVEL_STG are the two relevant tables available in the Manage System Configurations screen. It is possible to add/edit/delete rows in these tables. Information about the alternate hierarchy types and levels must be provided through these tables. Then, the alternate hierarchy jobs must be executed to generate the alternate hierarchies in AIF. The following table lists the relevant jobs.

Table 14-13 Alternate Hierarchy Jobs

JobName	Description	RMSBatch
RSE_ALT_HIER_SETUP_START_JOB	Start Job for Alternate Hierarchy Type Setup	rse_process_state_update.ksh
RSE_ALT_HIER_LOAD_JOB	Setup alternate hierarchy types by loading data from RSE_ALT_HIER_TYPE_STG table	rse_alt_hier_load.ksh
RSE_ALT_HIER_SETUP_END_JOB	End Job for Alternate Hierarchy Type Setup	rse_process_state_update.ksh
RSE_ALT_HIER_LOAD_ADHOC_JOB	Adhoc job to Setup alternate hierarchy types by loading data from RSE_ALT_HIER_TYPE_STG table	rse_alt_hier_load.ksh
RSE_ALT_LOC_HIER_START_JOB	Start Job for Alternate Location Hierarchy Load	rse_process_state_update.ksh
RSE_ALT_LOC_HIER_SRC_XREF_LOAD_JOB	Load the RSE_LOC_SRC_XREF table with alternate location hierarchy data	rse_alt_loc_hier_process.ksh
RSE_ALT_LOC_SRC_XREF_DUP_CHECK_JOB	Check for RSE_ALT_LOC_SRC_XREF column values that were either modified or for which rows were bypassed to avoid duplicate keys on index errors	rse_alt_loc_src_xref_dup_check.ksh
RSE_ALT_LOC_HIER_LOAD_JOB	Load the RSE_LOC_HIER table with alternate location hierarchy data	rse_alt_loc_hier_load.ksh
RSE_ALT_LOC_HIER_TC_LOAD_JOB	Load the RSE_LOC_HIER_TC table with alternate location hierarchy data	rse_alt_loc_hier_tc_load.ksh
RSE_ALT_LOC_HIER_DH_LOAD_JOB	Load the RSE_LOC_HIER_DH table with alternate location hierarchy data	rse_alt_loc_hier_dh_load.ksh
RSE_ALT_LOC_HIER_END_JOB	End Job for Alternate Location Hierarchy Load	rse_process_state_update.ksh
RSE_ALT_LOC_HIER_SRC_XREF_LOAD_ADHOC_JOB	Adhoc job to Load the RSE_LOC_SRC_XREF table with alternate location hierarchy data	rse_alt_loc_hier_process.ksh

Table 14-13 (Cont.) Alternate Hierarchy Jobs

JobName	Description	RMSBatch
RSE_ALT_LOC_SRC_XREF_DUP_CHECK_ADHOC_JOB	Adhoc job to Check for RSE_ALT_LOC_SRC_XREF column values that were either modified or for which rows were bypassed to avoid duplicate keys on index errors	rse_alt_loc_src_xref_dup_check.ksh
RSE_ALT_LOC_HIER_LOAD_ADHOC_JOB	Adhoc job to Load the RSE_LOC_HIER table with alternate location hierarchy data	rse_alt_loc_hier_load.ksh
RSE_ALT_LOC_HIER_TC_LOAD_ADHOC_JOB	Adhoc job to Load the RSE_LOC_HIER_TC table with alternate location hierarchy data	rse_alt_loc_hier_tc_load.ksh
RSE_ALT_LOC_HIER_DH_LOAD_ADHOC_JOB	Adhoc job to Load the RSE_LOC_HIER_DH table with alternate location hierarchy data	rse_alt_loc_hier_dh_load.ksh
RSE_ALT_PROD_HIER_START_JOB	Start Job for Alternate Product Hierarchy Load	rse_process_state_update.ksh
RSE_ALT_PROD_HIER_SRC_XREF_LOAD_JOB	Load the RSE_PROD_SRC_XREF table with alternate product hierarchy data	rse_alt_prod_hier_process.ksh
RSE_ALT_PROD_SRC_XREF_DUP_CHECK_JOB	Check for RSE_ALT_PROD_SRC_XREF column values that were either modified or for which rows were bypassed to avoid duplicate keys on index errors	rse_alt_prod_src_xref_dup_check.ksh
RSE_ALT_PROD_HIER_LOAD_JOB	Load the RSE_PROD_HIER table with alternate product hierarchy data	rse_alt_prod_hier_load.ksh
RSE_ALT_PROD_HIER_TC_LOAD_JOB	Load the RSE_PROD_HIER_TC table with alternate product hierarchy data	rse_alt_prod_hier_tc_load.ksh
RSE_ALT_PROD_HIER_DH_LOAD_JOB	Load the RSE_PROD_HIER_DH table with alternate product hierarchy data	rse_alt_prod_hier_dh_load.ksh
RSE_ALT_PROD_HIER_END_JOB	End Job for Alternate Product Hierarchy Load	rse_process_state_update.ksh
RSE_ALT_PROD_HIER_SRC_XREF_LOAD_ADHOC_JOB	Adhoc job to Load the RSE_PROD_SRC_XREF table with alternate product hierarchy data	rse_alt_prod_hier_process.ksh
RSE_ALT_PROD_SRC_XREF_DUP_CHECK_ADHOC_JOB	Adhoc job to Check for RSE_ALT_PROD_SRC_XREF column values that were either modified or for which rows were bypassed to avoid duplicate keys on index errors	rse_alt_prod_src_xref_dup_check.ksh
RSE_ALT_PROD_HIER_LOAD_ADHOC_JOB	Adhoc job to Load the RSE_PROD_HIER table with alternate product hierarchy data	rse_alt_prod_hier_load.ksh

Table 14-13 (Cont.) Alternate Hierarchy Jobs

JobName	Description	RMSBatch
RSE_ALT_PROD_HIER_TC_LO AD_ADHOC_JOB	Adhoc job to Load the RSE_PROD_HIER_TC table with alternate product hierarchy data	rse_alt_prod_hier_tc_load.ksh
RSE_ALT_PROD_HIER_DH_LO AD_ADHOC_JOB	Adhoc job to Load the RSE_PROD_HIER_DH table with alternate product hierarchy data	rse_alt_prod_hier_dh_load.ksh

To perform forecasting in AIF using the created alternate location or product hierarchy, the appropriate hierarchy types need to be selected while creating a new run type. See [Using Alternate Hierarchy to Set Up Run Types](#) for more details.

Any existing alternate hierarchies must be deleted first if you are going to restructure them with different levels and mappings for the same hierarchy type. Attempting to restructure a hierarchy without deleting it can result in data issues and job failures. If you need to delete an alternate hierarchy, perform the following steps.

1. Update the delete flag to **Y** on the row in the RSE_ALT_HIER_TYPE_STG table corresponding to the alternate hierarchy being deleted.
2. Delete all the level rows from the RSE_ALT_HIER_LEVEL_STG table corresponding to the alternate hierarchy being deleted.
3. Run the processes/jobs to rebuild the alternate hierarchies, in the same way you previously did to build those.

If you are working with standalone jobs, then either use RSE_MASTER_ADHOC_JOB to run the hierarchy loads again (using -X flag for the alternate hierarchy load, and others as needed) or use the adhoc jobs listed in the table above. Also, for batch runs, the jobs listed in the table above will handle the deletion. Note that you cannot delete and rebuild the same hierarchy in the same batch cycle or standalone process.

The alternate hierarchy batch or adhoc jobs will do the following cleanup:

1. The delete flag in the RSE_ALT_HIER_TYPE table will be set to **Y**.
2. The corresponding row in the RSE_HIER_TYPE table will be deleted.
3. The corresponding rows in the RSE_ALT_HIER_LEVEL table will be deleted.
4. The corresponding rows in the RSE_HIER_LEVEL table will be deleted.
5. The corresponding partition in the RSE_LOC_HIER and/or RSE_PROD_HIER tables will be dropped.
6. The corresponding partition in the RSE_LOC_HIER_DH and/ or RSE_PROD_HIER_DH tables will be dropped.
7. The corresponding partition in the RSE_LOC_HIER_TC and/or RSE_PROD_HIER_TC tables will be dropped.
8. The corresponding partition in the RSE_LOC_SRC_XREF and/or RSE_PROD_SRC_XREF tables will be dropped.

After alternate hierarchy batch or adhoc jobs run a second time, the remaining corresponding row in the RSE_ALT_HIER_TYPE table whose delete flag had been set to **Y** will then be deleted.

Using Alternate Hierarchy to Set Up Run Types

Each run type in AIF can potentially use a different location or product hierarchy. If an alternate hierarchy must be used in forecasting, then the first step is to add that hierarchy information to the table RSE_FCST_DFLT_RUN_TYPE_PARAM via the Manage System Configurations screen. If it is an alternate location hierarchy, then a new row must be added to the table under the PARAMETER_NAME = LOC_HIER_TYPE. Similarly, if it is an alternate product hierarchy, then a new row must be added to the table under the PARAMETER_NAME = PROD_HIER_TYPE. Following that, while creating a new run type in the Setup train stop within the Manage Forecast Configurations screen, the appropriate hierarchies must be selected in the Hierarchy Type - Merchandise and Hierarchy Type - Location drop-down lists. The Forecast Level - Merchandise and Forecast Level - Location drop-down list values will show up dynamically, based on the selected hierarchies.

Custom Jobs Through Innovation Workbench (IW)

Custom processes can be created through IW (see also [Loading Out-of-Stock and Outlier Indicators Through the RDX Interface](#) as an example). AIF provides nine in-built custom jobs that can be used for setting up some custom steps at various positions within the AIF Apps batch pipeline. The three pre-batch and the three post-batch jobs are positioned to run at the beginning and at the end of the AIF Apps batch, respectively. Additionally, for the forecast engine, the three pre-estim jobs are positioned to run between the RSPACE to RDX pre-forecast parameter export and the RDX to AIF pre-forecast parameter import. The following table lists the relevant jobs.

Table 14-14 Custom Jobs for IW Custom Processes

JobName	Description	RMSBatch
PMO_PRE_ESTIM_CUSTOM_S TART_JOB	Start job for PMO pre-estimation custom process	rse_process_state_update.ksh
PMO_PRE_ESTIM_CUSTOM_1_ JOB	PMO pre-estimation custom job 1	rse_custom_job.ksh
PMO_PRE_ESTIM_CUSTOM_2_ JOB	PMO pre-estimation custom job 2	rse_custom_job.ksh
PMO_PRE_ESTIM_CUSTOM_3_ JOB	PMO pre-estimation custom job 3	rse_custom_job.ksh
PMO_PRE_ESTIM_CUSTOM_E ND_JOB	End job for PMO pre-estimation custom process	rse_process_state_update.ksh
PMO_PRE_ESTIM_CUSTOM_1_ ADHOC_JOB	PMO pre-estimation custom ad hoc job 1	rse_custom_job.ksh
PMO_PRE_ESTIM_CUSTOM_2_ ADHOC_JOB	PMO pre-estimation custom ad hoc job 2	rse_custom_job.ksh
PMO_PRE_ESTIM_CUSTOM_3_ ADHOC_JOB	PMO pre-estimation custom ad hoc job 3	rse_custom_job.ksh
RSE_PRE_BATCH_CUSTOM_S TART_JOB	Start job for AIF Apps Pre-Batch custom process	rse_process_state_update.ksh
RSE_PRE_BATCH_CUSTOM_1_ JOB	AIF Apps Pre-Batch custom job 1	rse_custom_job.ksh
RSE_PRE_BATCH_CUSTOM_2_ JOB	AIF Apps Pre-Batch custom job 2	rse_custom_job.ksh

Table 14-14 (Cont.) Custom Jobs for IW Custom Processes

JobName	Description	RMSBatch
RSE_PRE_BATCH_CUSTOM_3_JOB	AIF Apps Pre-Batch custom job 3	rse_custom_job.ksh
RSE_PRE_BATCH_CUSTOM_END_JOB	End job for AIF Apps Pre-Batch custom process	rse_process_state_update.ksh
RSE_PRE_BATCH_CUSTOM_1_ADHOC_JOB	AIF Apps Pre-Batch custom ad hoc job 1	rse_custom_job.ksh
RSE_PRE_BATCH_CUSTOM_2_ADHOC_JOB	AIF Apps Pre-Batch custom ad hoc job 2	rse_custom_job.ksh
RSE_PRE_BATCH_CUSTOM_3_ADHOC_JOB	AIF Apps Pre-Batch custom ad hoc job 3	rse_custom_job.ksh
RSE_POST_BATCH_CUSTOM_START_JOB	Start job for AIF Apps Post-Batch custom process	rse_process_state_update.ksh
RSE_POST_BATCH_CUSTOM_1_JOB	AIF Apps Post-Batch custom job 1	rse_custom_job.ksh
RSE_POST_BATCH_CUSTOM_2_JOB	AIF Apps Post-Batch custom job 2	rse_custom_job.ksh
RSE_POST_BATCH_CUSTOM_3_JOB	AIF Apps Post-Batch custom job 3	rse_custom_job.ksh
RSE_POST_BATCH_CUSTOM_END_JOB	End job for AIF Apps Post-Batch custom process	rse_process_state_update.ksh
RSE_POST_BATCH_CUSTOM_1_ADHOC_JOB	AIF Apps Post-Batch custom ad hoc job 1	rse_custom_job.ksh
RSE_POST_BATCH_CUSTOM_2_ADHOC_JOB	AIF Apps Post-Batch custom ad hoc job 2	rse_custom_job.ksh
RSE_POST_BATCH_CUSTOM_3_ADHOC_JOB	AIF Apps Post-Batch custom ad hoc job 3	rse_custom_job.ksh

Here are the steps to get the custom jobs working:

1. In IW, create a package and procedure for the process that needs to be run as a custom job. After logging into Innovation Workbench -> Manage Workbench -> SQL Workshop, on the right-hand side click on the "Package" under "Create Object" and proceed with creating the package body, specification and create the procedure inside the package.
2. Using IW or the Manage System Config screen, modify the table RSE_CUSTOM_JOB_CFG and edit values for the following columns:
 - a. JOB_NAME: This value is not editable and indicates a placeholder where a custom process can be added.
 - b. PACKAGE_NAME: Enter the name of package that was created in IW.
 - c. PROCEDURE_NAME: Enter the name of procedure that was created in IW.
 - d. PROCEDURE_DESCR: Enter a description if desired.
 - e. RUN_TIME_LIMIT: The run time limit is 900 seconds by default. It can be changed to a smaller value but not to a larger value. If the custom process runs for longer than the value indicated in RUN_TIME_LIMIT when running as a part of the batch process, the custom process will stop and move on to the next job/process.
 - f. CONNECTION_TYPE: Valid values are LOW and MEDIUM. This value should almost always be LOW unless the job is supposed to run a process that would need multiple

threads. HIGH is not a valid value. If HIGH is entered, it will switch to LOW by default when the job runs.

- g. **ENABLE_FLG:** Set this value to Y to indicate that this job should be executed as part of the batch process.

Purge Forecast Run and Run Type Data

Forecast run data can be purged through both batch and ad hoc jobs. The parameters that control this purge process are **FCST_PURGE_RETENTION_DAYS** and **FCST_PURGE_ALL_RUNS** in the **RSE_CONFIG** table (**APPL_CODE** = RSE), available in the Manage System Configurations screen. The ad hoc job can take an input that acts as an override to the **FCST_PURGE_RETENTION_DAYS** value. When **FCST_PURGE_ALL_RUNS** = Y, then both the batch and ad hoc jobs delete data for all forecast runs, irrespective of the **FCST_PURGE_RETENTION_DAYS** value. Forecast run type data can be purged through the ad hoc job. The parameters that control this purge process are **FCST_TY_PURGE_RETENTION_DAYS** and **FCST_TY_PURGE_ALL_RUNS** in the **RSE_CONFIG** table (**APPL_CODE** = RSE), available in the Manage System Configurations screen. The ad hoc job can take an input that acts as an override to the **FCST_TY_PURGE_RETENTION_DAYS** value. When **FCST_TY_PURGE_ALL_RUNS** = Y, then the ad hoc job deletes data for all forecast run types, irrespective of the **FCST_TY_PURGE_RETENTION_DAYS** value. The following table lists the relevant jobs.

Table 14-15 Purge Jobs for Forecast Run and Run Type Data

JobName	Description	RMSBatch
RSE_PURGE_RUN_START_JOB	Start job for purging run data	rse_process_state_update.ksh
RSE_PURGE_RUN_JOB	Purge job for run data	rse_purge_run_data.ksh
RSE_PURGE_RUN_END_JOB	End job for purging run data	rse_process_state_update.ksh
RSE_PURGE_RUN_ADHOC_JOB	Ad hoc purge job for run data	rse_purge_run_data.ksh
RSE_PURGE_RUN_TYPE_ADHOC_JOB	Ad hoc purge job for run type data	rse_purge_run_type_data.ksh

Configurations in AIF to Generate Forecasts for FACT Measures

There are four FACT tables (**W_RTL_FACT1_PROD1_LC1_T1_F**, **W_RTL_FACT2_PROD2_LC2_T2_F**, **W_RTL_FACT3_PROD3_LC3_T3_F**, and **W_RTL_FACT4_PROD4_LC4_T4_F**) in RI that store the FACT measures data. The data from these FACT tables flow into the various critical AIF tables (**RSE_SLS_PR_LC_WK**, **RSE_SLS_PR_LC_CS_WK**, **RSE_SLS_PH_LC_WK_A**, and **RSE_INV_PR_LC_WK_A**) via **RSE_PR_LC_CAL_CDA**. The table **RSE_MD_CDA** programmatically stores the mapping between the FACT tables and the columns of **RSE_PR_LC_CAL_CDA**. The following table shows the jobs that will populate the various critical AIF tables from the FACT tables.

Table 14-16 Jobs to Populate the AIF Tables from FACT Tables

JobName	Description	RMSBatch
RSE_AGGREGATE_METADATA_LOAD_ADHOC_JOB	Ad hoc job to load aggregate measure metadata from RI to AIF (populates RSE_MD_CDA)	rse_aggregate_metadata_load.ksh

Table 14-16 (Cont.) Jobs to Populate the AIF Tables from FACT Tables

JobName	Description	RMSBatch
RSE_AGGREGATE_ACTUALS_LOAD_SETUP_JOB	Setup job to load aggregate measure actuals from RI to AIF	rse_aggregate_actuals_load_setup.ksh
RSE_AGGREGATE_ACTUALS_LOAD_PROCESS_JOB	Process job to load aggregate measure actuals from RI to AIF (populates RSE_PR_LC_CAL_CDA)	rse_aggregate_actuals_load_process.ksh
RSE_AGGREGATE_ACTUALS_LOAD_SETUP_ADHOC_JOB	Ad hoc setup job to load aggregate measure actuals from RI to AIF	rse_aggregate_actuals_load_setup.ksh
RSE_AGGREGATE_ACTUALS_LOAD_PROCESS_ADHOC_JOB	Ad hoc process job to load aggregate measure actuals from RI to AIF (populates RSE_PR_LC_CAL_CDA)	rse_aggregate_actuals_load_process.ksh
RSE_SLS_PR_LC_WK_AGGREGATE_MEAS_SETUP_JOB	Job for product at leaf level/location/week sales aggregate measure setup	rse_sls_pr_lc_wk_aggr_meas_setup.ksh
RSE_SLS_PR_LC_WK_AGGREGATE_MEAS_PROCESS_JOB	Job for product at leaf level/location/week sales aggregate measure process (populates RSE_SLS_PR_LC_WK and RSE_SLS_PR_LC_CS_WK)	rse_sls_pr_lc_wk_aggr_meas_process.ksh
RSE_SLS_PR_LC_WK_AGGREGATE_MEAS_SETUP_ADHOC_JOB	Ad hoc job for product at leaf level/location/week sales aggregate measure setup	rse_sls_pr_lc_wk_aggr_meas_setup.ksh
RSE_SLS_PR_LC_WK_AGGREGATE_MEAS_PROCESS_ADHOC_JOB	Ad hoc job for product at leaf level/location/week sales aggregate measure process (populates RSE_SLS_PR_LC_WK and RSE_SLS_PR_LC_CS_WK)	rse_sls_pr_lc_wk_aggr_meas_process.ksh
RSE_SLS_PH_LC_WK_AGGREGATE_MEAS_SETUP_JOB	Job for product at non-leaf level/location/week sales aggregate measure setup	rse_sls_ph_lc_wk_aggr_meas_setup.ksh
RSE_SLS_PH_LC_WK_AGGREGATE_MEAS_PROCESS_JOB	Job for product at non-leaf level/location/week sales aggregate measure process (populates RSE_SLS_PH_LC_WK_A)	rse_sls_ph_lc_wk_aggr_meas_process.ksh
RSE_SLS_PH_LC_WK_AGGREGATE_MEAS_SETUP_ADHOC_JOB	Ad hoc job for product at non-leaf level/location/week sales aggregate measure setup	rse_sls_ph_lc_wk_aggr_meas_setup.ksh
RSE_SLS_PH_LC_WK_AGGREGATE_MEAS_PROCESS_ADHOC_JOB	Ad hoc job for product at non-leaf level/location/week sales aggregate measure process (populates RSE_SLS_PH_LC_WK_A)	rse_sls_ph_lc_wk_aggr_meas_process.ksh
RSE_INV_PR_LC_WK_AGGREGATE_MEAS_SETUP_JOB	Job for product at leaf level/location/week inventory aggregate measure setup	rse_inv_pr_lc_wk_aggr_meas_setup.ksh
RSE_INV_PR_LC_WK_AGGREGATE_MEAS_PROCESS_JOB	Job for product at leaf level/location/week inventory aggregate measure process (populates RSE_INV_PR_LC_WK_A)	rse_inv_pr_lc_wk_aggr_meas_process.ksh

Table 14-16 (Cont.) Jobs to Populate the AIF Tables from FACT Tables

JobName	Description	RMSBatch
RSE_INV_PR_LC_WK_AGGR_MEAS_SETUP_ADHOC_JOB	Adhoc job for product at leaf level/location/week inventory aggregate measure setup	rse_inv_pr_lc_wk_aggr_meas_setup.ksh
RSE_INV_PR_LC_WK_AGGR_MEAS_PROCESS_ADHOC_JOB	Ad hoc job for product at leaf level/location/week inventory aggregate measure process (populates RSE_INV_PR_LC_WK_A)	rse_inv_pr_lc_wk_aggr_meas_process.ksh

Complete the following steps in Manage System Configurations screen to configure the generation of forecasts for FACT measures.

1. Configure data in RSE_FCST_DFLT_RUN_TYPE_PARAM.

The pre-configured measures for forecasting are specified in RSE_FCST_DFLT_RUN_TYPE_PARAM under the PARAMETER_NAME = FORECAST_MEASURE. To configure a new forecast measure (for example, Shrink Qty), the user must insert the values in RSE_FCST_DFLT_RUN_TYPE_PARAM using the UI.

For example, a row with the following values must be inserted.

(PARAMETER_NAME, PARAMETER_VALUE, DESCR, UI_PARAMETER_LABEL, DISPLAY_ORDER, CURRENT_FLG, DELETE_FLG)

('FORECAST_MEASURE', 'SHRINK_QTY', 'Shrink Qty', 'Shrink Qty', '13', 'Y', 'N')

2. Configure data in RSE_MD_FACT_COLUMN_MAP.

To generate a forecast for FACT measures, data must flow into the PMO_ACTIVITIES table. If the FACT measure is one of the forecast measures that are available in the GA forecast engine, then the configuration must be done such that the appropriate named column in PMO_ACTIVITIES is used in the mapping. The GA measures can be found in RSE_FCST_DFLT_RUN_TYPE_PARAM under the PARAMETER_NAME = FORECAST_MEASURE.

If the FACT measure is not one of the GA measures, then the configuration must be done such that one of the flex measure columns in PMO_ACTIVITIES is used in the mapping.

For example, if user wants to forecast Shrink Qty, the FACT measure SHRINK_QTY must be loaded to one of the FLEX columns in PMO_ACTIVITIES. The following row must be inserted into RSE_MD_FACT_COLUMN_MAP using the UI.

(SOURCE_TABLE, SOURCE_COLUMN, TARGET_TABLE, TARGET_COLUMN)

('W_RTL_FACT1_PROD1_LC1_T1_F', 'SHRINK_QTY', 'PMO_ACTIVITIES', 'FLEX_MEAS_VALUE1')

3. Configure data in RSE_FCST_RUN_TYPE_MTHD_PARAM.

The forecast engine supports a few different forecast methods, and each method can be used to forecast certain measures based on certain data sources. This mapping is configured in RSE_FCST_RUN_TYPE_MTHD_PARAM. FACT measures can be forecasted using Automatic Exponential Smoothing. Note that all the required data for the method that is being used must be provided via the interface for FACT measures.

For example, to forecast Shrink Qty using Automatic Exponential Smoothing, the following row must be inserted in RSE_FCST_RUN_TYPE_MTHD_PARAM using the UI.

(FORECAST_METHOD, FORECAST_MEASURE, DESCR,
FORECAST_MEASURE_UI_LABEL, FORECAST_METHOD_UI_LABEL,
CURRENT_FLG, DATA_SOURCE)

('AUTO_ES', 'SHRINK_QTY', 'Shrink Qty', 'Shrink Qty', 'Automatic Exponential
Smoothing', 'Y', 'STORE_SALES')

The following table shows the jobs that must be run to aggregate FACT measures data (as required) and populate PMO_ACTIVITIES.

Table 14-17 Jobs to Populate PMO_ACTIVITIES Using FACT Data

JobName	Description	RmsBatch
PMO_ACTIVITY_AGGR_MEAS_SETUP_JOB	PMO aggregate measures load setup job	pmo_activity_aggr_meas_setup.ksh
PMO_ACTIVITY_AGGR_MEAS_PROCESS_JOB	PMO aggregate measures load process job	pmo_activity_aggr_meas_process.ksh
PMO_ACTIVITY_AGGR_MEAS_SETUP_ADHOC_JOB	PMO aggregate measures load setup adhoc job	pmo_activity_aggr_meas_setup.ksh
PMO_ACTIVITY_AGGR_MEAS_PROCESS_ADHOC_JOB	PMO aggregate measures load process adhoc job	pmo_activity_aggr_meas_process.ksh

Workflow Example for Generating Forecast for FACT Measures

Here is a set of sample steps for generating forecast for MFP using Automatic Exponential Smoothing as the forecast method, Other Gross Sales Amount as the forecast measure, and Fact Measures as the Data Source.

1. Update the parameter SWITCH_TO_TXN_LOAD_DATE in the RSE_CONFIG table (APPL_CODE = RSE), available in the Manage System Configurations screen, to a date that denotes the end of FACT data load. If data is always going to come to AIF via the RI FACT tables, then set this date to something very far in the future.
2. In the RSE_FCST_DFLT_RUN_TYPE_PARAM table, available in the Manage System Configurations screen, set CURRENT_FLG = Y for the row where PARAMETER_NAME = DATA_SOURCE and PARAMETER_VALUE = FACT_MEASURES.
3. In the RSE_FCST_DFLT_RUN_TYPE_PARAM table, verify that there is already a row for PARAMETER_NAME = FORECAST_MEASURE and PARAMETER_VALUE = OTHER_GROSS_AMOUNT, because Other Gross Sales Amount is a GA forecast measure. Otherwise, add a row for the forecast measure as necessary.
4. Suppose the FACT data is available in the RI table W_RTL_FACT3_PROD3_LC3_T3_F in the column FLEX2_NUM_VALUE. For generating forecast, the data needs to be populated in the AIF table PMO_ACTIVITIES in the appropriate column representative of the forecast measure, which is OTHER_GROSS_SALES_AMOUNT in this case. In the RSE_MD_FACT_COLUMN_MAP table, available in the Manage System Configurations screen, add a row with the entries being SOURCE_TABLE = W_RTL_FACT3_PROD3_LC3_T3_F, SOURCE_COLUMN = FLEX2_NUM_VALUE, TARGET_TABLE = PMO_ACTIVITIES, and TARGET_COLUMN = OTHER_GROSS_SALES_AMOUNT. If the forecast measure is non-GA, then use one of the flex measure columns in PMO_ACTIVITIES.
5. Specific combinations of data source, forecast method, and forecast measure is supported by the AIF forecast engine. In the RSE_FCST_RUN_TYPE_MTHD_PARAM table, available in the Manage System Configurations screen, add a row with the entries being DATA_SOURCE = FACT_MEASURES, FORECAST_METHOD = AUTO_ES, FORECAST_METHOD_UI_LABEL = Automatic Exponential Smoothing,

FORECAST_MEASURE = OTHER_GROSS_AMOUNT,
FORECAST_MEASURE_UI_LABEL = Normalized SIs Val, DESCR = Normalized SIs Val,
and CURRENT_FLG = Y. Note that the UI label and description entries can be set to
anything suitable.

6. Run the RSE_AGGREGATE_METADATA_LOAD_ADHOC_JOB (under the process RSE_LOAD_AGGREGATE_FACT_DATA_ADHOC_PROCESS) to populate the table RSE_MD_CDA. This table shows the mapping between the columns of the RI fact tables and RSE_PR_LC_CAL_CDA. For this example, the entries in RSE_MD_CDA should auto-populate as follows: SRC_ENTITY_NAME = W_RTL_FACT3_PROD3_LC3_T3_F, SRC_COLUMN_NAME = FLEX2_NUM_VALUE, and TGT_COLUMN_NAME = ATTR_NUM_VALUE_1. The TGT_COLUMN_NAME can be any column in RSE_PR_LC_CAL_CDA.
7. Run the RSE_AGGREGATE_ACTUALS_LOAD_ADHOC_JOB (under the process RSE_LOAD_AGGREGATE_FACT_DATA_ADHOC_PROCESS) to populate the table RSE_PR_LC_CAL_CDA. Ensure that the data is synced up between the source table (in this case W_RTL_FACT3_PROD3_LC3_T3_F) and RSE_PR_LC_CAL_CDA. The RSE_PROC_STAT table should also reflect a row under PROC_NAME = RSE_AGGR_MEASURE_ETL-W_RTL_FACT3_PROD3_LC3_T3_F, where the UPDATE_DT shows the date when the job was last executed and the LAST_PROCESSED_HIER_ID shows the last week of fact data loaded.
8. In Manage Forecast Configurations train stop 1 Setup, create a run type with Data Source = Fact Measures, Forecast Method = Automatic Exponential Smoothing, Forecast Measure = Normalized SIs Val, and using the appropriate merchandise/location hierarchy types and forecast levels. Following the run type creation, if needed, complete the data aggregation step. This will populate the PMO_ACTIVITIES table at the appropriate AGGR_LVL_KEY. Ensure that the data is synced up between RSE_PR_LC_CAL_CDA and PMO_ACTIVITIES. The RSE_PROC_STAT table should also reflect a row under PROC_NAME = PMO_ACTIVITY_AGGR_MEAS.
9. If in-season (Bayesian) forecast is required, then sales plan also needs to be loaded into AIF. For this example, suppose sales plan number 3 is used. In the Manage Master Data screen, edit the row corresponding to sales plan number 3 to set up the same merchandise/location hierarchy types and hierarchy levels as used in the run type. For sales plan number 3, the RI table W_RTL_PLAN3_PROD3_LC3_T3_F needs to store the plan data in the appropriate column representative of the forecast measure, which is FLEX2_NUM_VALUE in this case (for the forecast measure Other Gross Sales Units, the column should be FLEX1_NUM_VALUE). Run the RSE_FCST_SALES_PLAN_LOAD_ADHOC_JOB (under the process RSE_FCST_SALES_PLAN_LOAD_ADHOC_PROCESS) to populate the tables RSE_FCST_SALES_PLAN_HDR and RSE_FCST_SALES_PLAN_DTL (column SLSOTH_RTL_AMT). Ensure that the data is synced up between the source table (in this case W_RTL_PLAN3_PROD3_LC3_T3_F) and RSE_FCST_SALES_PLAN_DTL.
10. In Manage Forecast Configurations train stop 2 Test, under the appropriate run type create different forecast runs to test the impact of the parameters and validate the results in the table RSE_FCST_MFP. For this example, the column OTH_FCST_AMT should be populated.
11. In Manage Forecast Configurations train stop 3 Manage, for the appropriate run type update the parameter values for usage in the batch forecast runs, activate the run type, and enable auto-approve. After this ensure that the data is synced up between the source table (in this case W_RTL_FACT3_PROD3_LC3_T3_F), RSE_PR_LC_CAL_CDA, and PMO_ACTIVITIES for the appropriate AGGR_LVL_KEY. To sync up RSE_PR_LC_CAL_CDA, first delete the row from RSE_PROC_STAT table under PROC_NAME = RSE_AGGR_MEASURE_ETL-W_RTL_FACT3_PROD3_LC3_T3_F and run RSE_AGGREGATE_METADATA_LOAD_ADHOC_JOB. To sync up

PMO_ACTIVITIES, execute the relevant jobs (PMO_ACTIVITY_AGGR_MEAS_SETUP_ADHOC_JOB and PMO_ACTIVITY_AGGR_MEAS_PROCESS_ADHOC_JOB) in PMO_ACTIVITY_LOAD_ADHOC_PROCESS with the necessary parameters.

12. In Manage Forecast Configurations train stop 4 Map, map the appropriate run type to MFP.
13. Run RSE_CREATE_MFP_BATCH_RUN_PROC_ADHOC_JOB and RSE_MFP_FCST_BATCH_PROCESS_ADHOC_JOB (under the process RSE_MFP_FCST_BATCH_RUN_ADHOC_PROCESS) to execute a forecast batch.
14. Run RSE_MFP_FCST_EXPORT_ADHOC_JOB (under the process RSE_FCST_EXPORT_ADHOC_PROCESS) to export the forecast to the RDX table RSE_FCST_DEMAND_EXP. For this example, the column OTH_SLS_AMT should be populated.

Data Requirements for AIF

AI Foundation has a wide range of capabilities and solutions, each with its own specific data needs. While the files themselves are all common across the Retail Analytics and Planning solutions, which fields you provide data for require careful consideration. This section covers the four most critical data areas that must be discussed with customers as part of requirements and design. This data will directly impact the results across your implementation and determine which RAP features you will be able to benefit from, both now and in the future. This is not a replacement for the Implementation guide or the Interface guide, as those provides more detailed requirements for each cloud service. Refer to the detailed documentation before you begin the development of the data files.

Data Definitions for Core Fact Files

This section describes the data definitions for core fact files.

SALES.csv

Sales data represents sales transactions resulting from customer purchases and returns. Sales data can be at varying levels of detail, depending on the solutions used, ranging from item/loc/week summary data to the granular transaction-line details. This is the core fact used across all modules and it has many variations, as defined below. How you define your sales data will affect both current and future implementations of any AIF solutions, so you must consider your product roadmap when deciding on the level of detail to provide here. If you provide pre-aggregated data, for example, then it will prevent you from using modules that require transaction data without doing a re-implementation of your data loads and integrations. Any additional effort spent on the data up front can result in significant savings on implementation costs for future projects.

This section provides a high-level summary of the main considerations for the file. More information about details and requirements is available in the “Sales Data Requirements” section of the Data File Generation chapter of *Oracle Retail Analytics and Planning Implementation Guide*.

Minimum Data (Forecasting)

The minimum data for forecasting has several types, depending on the use case. From least to most detailed, they are:

1. Item/loc/week or item/loc/day aggregated sales
2. Item/loc/week or item/loc/day transactional sales

3. Item/loc/week or item/loc/day promotion-linked transactional sales
4. Item/loc/week or item/loc/day customer-linked transactional sales

If you are only implementing basic Inventory Planning Optimization Demand Forecasting (IPO-DF) or Lifecycle Pricing Optimization (LPO) forecasting and do not require transaction level details, then you can simply provide pre-summed item/loc/day or week sales totals. Detailed forecast method breakdowns are provided in later sections to clarify requirements by forecast type. If you only want to forecast the units, then you can simply provide SLS_QTY for the gross sales units. SLS_TRX_ID is defaulted to a unique value to differentiate the rows, as it is part of the table key. You might only have a single row per item/loc/week/retail type with one value of SLS_TRX_ID per data intersection.

```
ITEM,ORG_NUM,DAY_DT,MIN_NUM,RTL_TYPE_CODE,SLS_TRX_ID,SLS_QTY,LOC_CURR_CODE,DOC_CURR_CODE
```

```
1234781,2072,20210131,0000,R,12345678,112,USD,USD
```

If you are forecasting based on sales amounts or net sales/returns, then you can also include the additional columns for those values, which must still be at a day or week aggregated level.

If you need to perform more advanced forecasting methods (Halo and Cannibalization feature) or you are using transaction-level AIF applications (Customer Decision Trees, Customer Segmentation), then you *must* provide the actual sales transaction information, not the aggregated values. In this instance, the SLS_TRX_ID is a critical field that must represent the true POS transaction identifier to uniquely identify one customer basket/sale.

```
ITEM,ORG_NUM,DAY_DT,MIN_NUM,RTL_TYPE_CODE,SLS_TRX_ID,SLS_QTY,SLS_AMT_LCL,RET_QTY,RET_AMT_LCL,LOC_CURR_CODE,DOC_CURR_CODE
```

```
1234781,2072,20210131,1700,R,21013172971,1,124,0,0,USD,USD
1234799,2072,20210131,1700,R,21013172971,1,55,0,0,USD,USD
1234884,2072,20210131,1700,R,21013172971,2,70,0,0,USD,USD
1235584,2072,20210131,1700,R,21013172971,0,0,1,99,USD,USD
```

This represents a single customer purchase in which the customer bought three items and returned one. The SLS_TRX_ID is the same for all transaction lines belonging to this one customer sale. If your POS system does not have a unique transaction ID by itself, then you may have to concatenate other parts on top of it (that is, take the POS tran number + loc number + register + date). A single day's sales file to AIF will have many such lines for the same item/loc for every sale or return transaction that happened that day. This must match with your POS system. Do not default the SLS_TRX_ID to some other value or the AIF results will not be accurate.

If you are using any of the promotion components of forecasting, such as promotion lift or estimation, then the sales data must be linked to the specific promotion IDs that occurred on them. This comes as two fields for the promotion ID and individual offer/component ID of the discount or price change in the promotion. One transaction line can have multiple promotions on it, which come as nearly identical lines of sales that differ only by promotion IDs.

```
ITEM,ORG_NUM,DAY_DT,MIN_NUM,RTL_TYPE_CODE,SLS_TRX_ID,PROMO_ID,PROMO_COMP_ID,SLS_QTY,SLS_AMT_LCL,RET_QTY,RET_AMT_LCL,LOC_CURR_CODE,DOC_CURR_CODE
```

```
1234781,2072,20210131,1700,P,21013172971,2498,12,1,124,0,0,USD,USD
1234799,2072,20210131,1700,P,21013172971,2498,12,1,55,0,0,USD,USD
1234799,2072,20210131,1700,P,21013172971,4693,111,1,55,0,0,USD,USD
```

```
1235584, 2072, 20210131, 1700, R, 21013172971, -2, -3, 0, 0, 1, 99, USD, USD
1234781, 2072, 20210131, 1800, R, 21013173592, -2, 3, 2, 248, 0, 0, USD, USD
```

This example shows two different promotions, and one line item that had two promotions on it in the same transaction. Note that non-promotional sales use the default values of -2 and -3, respectively. These are the standard values the platform uses when a promotion is not known or not used. If you are providing promotion identifiers, you will also be expected to provide other files relating to promotions, such as PROMOTION.csv. The *Oracle Retail Analytics and Planning Cloud Services Data Interfaces* provides the complete details on other promotion files and their requirements.

The last of the minimum required datasets is customer-linked transaction data, required if you are also using AIF modules such as CDT, DT, Segmentation, or segment-based forecasting/optimizations. You must provide the unique customer identifier for each sales transaction that is linked to a known customer in the source system.

```
ITEM, ORG_NUM, DAY_DT, MIN_NUM, RTL_TYPE_CODE, SLS_TRX_ID, CUSTOMER_NUM, SLS_QTY, SLS_
AMT_LCL, RET_QTY, RET_AMT_LCL, LOC_CURR_CODE, DOC_CURR_CODE
```

```
1234781, 2072, 20210131, 1700, R, 21013172971, 112193, 1, 124, 0, 0, USD, USD
1234884, 2072, 20210131, 1700, R, 21013172971, 112193, 2, 70, 0, 0, USD, USD
1235584, 2072, 20210131, 1700, R, 21013172971, 112193, 0, 0, 1, 99, USD, USD
1234781, 2072, 20210131, 1800, R, 21013173592, 33986, 2, 248, 0, 0, USD, USD
```

Field-Level Data Variations

As mentioned previously, DAY_DT can be daily dates or week-ending dates, depending on the requirements. If you are providing weekly files, it is fine to still provide the daily DAY_DT values for the days the transaction occurred, but all the data is provided in one week-ending file and no data is provided on other days of the week. (This is what is meant by *item/loc/week transaction file*: it is a weekly file with accumulated daily transactions.) When using actual transaction level data, however, it is best to provide the data every day due to the data volumes involved. Providing one file only at end of week can cause the weekly batch processing to be too long.

SLS_TRX_ID can be defaulted to a random/unique value if aggregated data is being used, but it must be the true POS customer transaction identifier if transaction level data is being used, as defined in the minimum requirement. It must always be provided as it is part of the primary key on the table.

RTL_TYPE_CODE allows values of R/P/C/O for regular/promotion/clearance/other sales, but not all types are required for all solutions. In some simple IPO-DF implementations, only Regular/R sales can be used to start if promotions/clearances do not matter to you. Note that you must delete and reload the sales history later if you need to change the retail types.

IT_SEQ_NUM is a situational column that you can provide if you are sending raw POS transactions directly from the source. This value uniquely identifies line numbers on a transaction when the same item could be scanned repeatedly. A unique value is required here in such cases or the duplicate rows will be discarded.

REVISION_NUM is a situational column that helps to track the number of revisions made to the same transaction over time. If you are providing aggregated data, then this field is not required. If you are providing transaction data with meaningful SLS_TRX_ID values, then you can also provide REVISION_NUM starting from one to track the number of revisions done to a transaction, as compared to the source system. Each time you post an update to a transaction, increment the number by one, and this will be updated on the transaction record.

SLS_PROFIT_AMT_LCL and RET_PROFIT_AMT_LCL may be relevant to you as additional measure columns if you require any measurement of cost. There are no cost fields in the SALES.csv file; you are instead expected to provide the sales profit/margin amounts associated with the transaction. The platform derives cost-related measures internally by subtracting the profit from the total sales amount when needed, such as when populating Sales Cost for RPASCE.

RECEIPT.csv

Inventory receipts data refers to any movement of stock into a location, which can be the result of any allocation, transfer, or PO shipment accepted into that location. Receipts data for AIF applications drives the first/last receipt dates by item/location as described in the data requirements in a later section. The quantity and type of receipts do not matter to forecasting, they matter only to RI reporting and applications that directly use receipts (IPO-IO, MFP, AP).

Receipt dates indicate that the first or last time that a receipt transaction occurred against an item/location is being calculated. You do not directly provide these calculated measures. You provide the transactions when receipts are occurring, and the platform will maintain calculations for first and last receipt dates. For example, a new item gets a receipt on 5/3/2023, so you will post a receipt transaction with this date for that item/location. During the data load, the RAP interface will take the inventory update, which should reflect the change to stock on hand, and the receipt update, which indicates the date that the receipt occurred, and calculate that the first receipt date and last receipt date should both be 5/3/2023 now. If another receipt arrives later, the new inventory and receipt records will be used to calculate the last receipt date to be that new date, while the first receipt date remains unchanged.

This document provides a high-level summary of the main considerations for the file. More details and requirements are available in the “Receipts Data Requirements” section of the Data File Generation chapter of *Oracle Retail Analytics and Planning implementation Guide*.

Minimum Data (Forecasting)

For forecasting at item/loc/week, it is sufficient to provide a single receipt record for any item/loc having a receipt during that week so that the first and last receipt date is updated accordingly. You do not need every possible receipt between the first and last if you are only interested in establishing the first/last receipt dates for an item/loc. You can leave out in-between records as long as you still align the receipt and inventory load time ranges. The type and quantity do not matter to forecasting. Forecasting only needs to know that inventory entered that location for that item. For example, this might be the only entry for an item/loc/week:

```
ITEM,ORG_NUM, DAY_DT, INVRCTYPE_CODE, LOC_CURR_CODE, DOC_CURR_CODE
```

```
30025,1183,20210123,20,USD,USD
```

Field-Level Data Variations

DAY_DT can be daily dates or week-ending dates. In either case, it is aggregated to week level for the purpose of weekly forecast usage. The data can only come at the end-of-week dates in batch, and all other days can have no data for this file. If you require daily data, then you can send updates every day or accumulate all the updates and provide one end-of-week file containing all the daily receipt activity. If you only send end-of-week data, then your inventory must likewise be end-of-week only, so the two files are aligned. Inventory and receipts must be aligned. It does not make sense for a receipt to occur but no inventory update to be posted. They must both be either daily files or weekly files.

The INVRCTYPE_CODE value only matters to RI, IPO-IO, and MFP. You can default it to some value (for example, 44-T) for all rows if not required in those applications. Type 20 (PO

receipts) typically happen at warehouses and are required to accurately represent WH receipts. Type 44~T are transfer receipts. This is most common for store locations as it is the receipt of any inventory as a result of a transfer between two locations. The source can be a WH or another store. Type 44~A is an optional sub-type for receipts that occurred due to allocations from the WH to the store. This type matters only to RI reporting at this time.

INVENTORY.csv

Inventory positions (that is, stock on hand) capture the current stock levels at all owned locations. Any time a transaction occurs involving stock on hand (sales, receipts, price changes, and so on.), an update to the inventory position must also occur to post the new stock level and related measures to AIF.

This section provides a high-level summary of the main considerations for the file. More details and requirements are available in the “Inventory Data Requirements” section of the Data File Generation chapter of *Oracle Retail Analytics and Planning implementation Guide*.

Minimum Data (Forecasting)

For forecasting at item/loc/week, it is sufficient to provide a single item/loc/week record for each item/loc on the week-ending date. The on-hand (INV_SOH_QTY) is required as it is used in base demand calculation. INV_IN_TRAN_QTY and INV_ON_ORD_QTY are not necessarily required, but will contribute to the total inventory quantity for forecasting purposes, and some applications, such as LPO and IPO-IO, will expect it, based on the solution requirements. The data must exactly match the receipt activity, as it is assumed any receipt will create a change to inventory. Conversely, any change to inventory must be the result of some transaction, if you are using the file such as a sale, adjustment, receipt, or RTV.

```
ITEM,ORG_NUM,DAY_DT,CLEARANCE_FLG,INV_SOH_QTY,LOC_CURR_CODE,DOC_CURR_CODE
```

```
30025,1183,20210123,N,5,USD,USD
```

```
30027,1112,20210123,N,0,USD,USD
```

Field-Level Data Variations

For history data, DAY_DT is always a week-ending date; it cannot be daily. Receipts history can be daily or weekly, and any receipt dates will be used by the weekly history loads to determine first/last receipt dates.

For ongoing batch files, the data can come every day or end-of-week only. If the data comes weekly, then it must be the final week-ending positions of all item/locs that had any change during the week. You must only send one record per item/loc/week with the final ending stock on hand. DAY_DT cannot be different on different rows in the file; it must always be the current day on all rows or the week-ending date on all rows. Because of the first/last receipt date calculations discussed regarding the receipts file, the RECEIPT and INVENTORY files must be in sync for ongoing loads. That is, both must send data daily or weekly, and you must not mix them. If you do not care about the receipt date calculations because you are not using those forecasting methods and never will, then this requirement can be ignored and you can send daily receipts but weekly inventory. However, this is not recommended.

The CLEARANCE_FLG is not a part of the primary key in the data. You cannot have multiple records per item/loc with different flags. However, it is a required field, as the inventory status (regular or clearance) is necessary for many of the applications. If you are only implementing an application such as IPO-DF and do not plan by clearance/non-clearance separately, you can default the value to N on all rows.

The data must include zero-balance INV_SOH_QTY rows in some cases. For example, if a receipt occurred but the item immediately sold out, an INV_SOH_QTY=0 record must still be

provided for that day or week because it is required to link with the receipt activity. If an inventory balance moved from non-zero to zero in that day or week, the zero balance record is also required to come once so that there is an inventory level update indicating it went out of stock. Other kinds of changes in the source can also generate an inventory record update even for a zero balance, such as changes to the unit cost/retail price requiring an update to some fields on this interface. Refer to the *Oracle Retail Analytics and Planning Implementation Guide* for additional details on building the inventory file and zero-balance needs in history and ongoing data.

If you are familiar with the RMFCS (Merchandising Foundation CS), the way inventory is extracted from that system is by looking at *all* transaction activity for the day and pulling inventory for every item/loc with any transactions of any kind. Even unrelated transaction types will trigger inventory pulls to ensure the daily positions in RAP are up to date for all fields.

PRICE.csv

The price data represents price or cost change events for an item/location. The data is only for executed events, not future planned events. Typically, these will include the same price events that are pushed out to stores to update the selling prices for all items.

This section provides a high-level summary of the main considerations for the file. More details and requirements are available in the “Price Data Requirements” section of the Data File Generation chapter of *Oracle Retail Analytics and Planning implementation Guide*.

Minimum Data (Forecasting)

For forecasting at item/loc/week, it is sufficient to provide any new prices and price changes that are effective on the week-ending date per item/loc. Both the SELLING_UNIT_RTL_AMT_LCL and BASE_COST_AMT_LCL must be provided on all the records unless you are only implementing SPO or IPO-DF, and then only SELLING_UNIT_RTL_AMT_LCL is required.

ITEM, ORG_NUM, DAY_DT, PRICE_CHANGE_TRAN_TYPE,
SELLING_UNIT_RTL_AMT_LCL, BASE_COST_AMT_LCL, LOC_CURR_CODE, DOC_CURR_CODE

83074502, 2030, 20200208, 0, 103.76, 49.34, USD, USD

Field-Level Data Variations

DAY_DT can be daily or weekly. In either case, it is aggregated to week level for the purpose of weekly forecast usage. The history data can also be daily or weekly, which is different from inventory. Daily price changes are loaded from history files exactly as provided for the date they occurred. You only provide a record when a price change event happens for an item/loc.

For ongoing batch files, the data can come every day or only for end-of-week. If the data comes weekly, then it must be the final week-ending positions of all item/locs that had any change during the week. You must only send one record per item/loc/week with final ending price. DAY_DT cannot be different on different rows in the file; it must always be the current day on all rows or the week-ending date on all rows. If you know that multiple price changes will occur within one week, then you must send the updates every day in order to capture each price change as it happens. For example, if a promotion or clearance happens in the middle of the week and ends two days later, sending a weekly price only will not capture the price changes. You must only send the week-ending regular price. In this case, you must plan to send daily price updates.

PRICE_CHANGE_TRAN_TYPE must be managed correctly to capture the type of price changes for an item/loc. The typical flow of price changes that will occur for an item/location must be as follows:

1. New price/cost (PRICE_CHANGE_TRAN_TYPE=0)
2. Regular cost changes (PRICE_CHANGE_TRAN_TYPE=2)
3. Regular price changes (PRICE_CHANGE_TRAN_TYPE=4)
4. Promotional/temporary markdowns (PRICE_CHANGE_TRAN_TYPE=9)
5. Clearance markdowns (PRICE_CHANGE_TRAN_TYPE=8)
6. Price reset due to new selling cycle or season change (PRICE_CHANGE_TRAN_TYPE=0)

All item/locations must have an initial price when the item is created (type 0). An item/location can then have many changes with types 2/4/8/9 before eventually staying at a type of 8 (for a final markdown) or resetting to a type of 0 (if the item lifecycle must restart).

Data Requirements for Various Forecast Methods

This section describes the data requirements for various forecast methods.

SALES.csv

Forecast Method: Sales & Promo

This forecast method requires data from SALES.csv. Sales data at the item/location level is required to run this method. The requirement is to provide sales history for a minimum of two years, as well as on an ongoing basis.

1. When extreme exogenous situations occur, enough sales history data is required on either side of the event.
2. If you want to generate customer segment-level forecasts, then you must provide the customer-linked transactions sales data. The retailer does not have to provide the data aggregated to the segment-level but must provide the customer ID and customer segment mapping as part of customer segment interface.
3. If the user wants to forecast by net sales units, then sales returns must also be provided so that the system can calculate the net sales units. As a general rule, items that exhibit very low return rates (less than 10 percent) do not have to worry about providing this additional data.
4. Regular and Promotional Sales sent via this interface are used to generate forecasts on these measures.
5. Transaction level data is required for using the Halo and Cannibalization feature.

When the system is in production, the latest incremental sales data is obtained as part of the batch process.

Forecast Method: Life Cycle

This forecast method requires data from SALES.csv. Sales data at the item/location level is required to run this method. The requirement is to provide sales history for a minimum of one year and an additional one season, which for most retailers is close to 18 to 24 months, as well as on an ongoing basis. It is recommended to provide sales history for two years.

1. When extreme exogenous situations occur, enough sales history data is required on either side of the event.
2. If you want to generate customer segment-level forecasts, then you must provide the customer-linked transactions sales data. The retailer does not have to provide the data aggregated to the segment-level but must provide the customer ID and customer segment mapping as part of customer segment interface.

3. If the user wants to forecast by net sales units, then sales returns must also be provided so that system can calculate the net sales units. The returns data identifies the original location where the item was purchased and when it was purchased. This information is useful in returns analysis and in calculating the returns parameters.
4. Regular and Promotional Sales sent via this interface are used to generate forecasts on these measures. Note that LPO supports optimization on the Total Gross Sales Units or Total Net Sales Units with this forecasting method.
5. Transaction level data is required for using the Halo and Cannibalization feature.

When the system is in production, the latest incremental sales data is obtained as part of the batch process.

Forecast Method: Automatic Exponential Smoothing

This forecast method requires data from SALES.csv. Sales data at the item/location level is required to run this method. The requirement is to provide sales history for a minimum of two years, as well as on an ongoing basis.

1. When extreme exogenous situations occur, enough sales history data is required on either side of the event.
2. If the user wants to forecast returns, then this must also be provided.
3. Regular Sales, Promotional Sales, Clearance Sales, and Other Sales are also sent via this interface. These are required to generate forecasts on these measures.

When the system is in production, the latest incremental sales data is obtained as part of the batch process.

Forecast Method: Hierarchical Bayesian

This forecast method requires data from SALES.csv. Sales data at the item/location level is required to run this method. The requirement is to provide sales history for a minimum of one year, as well as on an ongoing basis. It is recommended to provide sales history for one year and an additional one season length, which for most retailers is close to 18 to 24 months.

1. When extreme exogenous situations occur, enough sales history data is required on either side of the event.
2. If you want to generate customer segment-level forecasts, then you must provide the customer-linked transactions sales data. The retailer does not have to provide the data aggregated to the segment-level but must provide the customer ID and customer segment mapping as part of customer segment interface.
3. If the user wants to forecast by net sales units, then sales returns must be provided as well so that the system can calculate the net sales units.
4. This method only supports a forecast on the Total Gross Sales Units and Total Net Sales Units. Note that LPO supports optimization on the Total Gross Sales Units or Total Net Sales Units with this forecasting method.

When the system is in production, the latest incremental sales data is obtained as part of the batch process.

RECEIPT.csv

Forecast Method: Sales & Promo

This forecast method does not require data from RECEIPT.csv.

Forecast Method: Life Cycle

This forecast method requires data from RECEIPT.csv. The requirement is to provide the snapshot of the first and last inventory receipt dates for all active items, as well as on an ongoing basis. It is recommended to provide the first and last inventory receipt dates for all items, at least for the most recent two quarters. The first and last inventory receipt dates are used to calculate the model start date, which helps assign the season code necessary to choose the correct seasonality curves. The RSE_FIRST_INV_RECEIPT_HIST_MAX_WEEKS parameter value in RSE_CONFIG table (APPL_CODE = RSE), available in the Manage System Configurations screen, controls the configuration that records which first inventory receipt dates older than this many weeks from the latest week with inventory will be excluded from model start date calculation.

Forecast Method: Automatic Exponential Smoothing

This forecast method does not require data from RECEIPT.csv.

Forecast Method: Hierarchical Bayesian

This forecast method does not require data from RECEIPT.csv.

INVENTORY.csv**Forecast Method: Sales & Promo**

This forecast method does not require data from INVENTORY.csv. In the absence of inventory data from the env, the PMO_AGGR_INVENTORY_DATA_FLG parameter value in RSE_CONFIG table (APPL_CODE = PMO) available in the Manage System Configurations screen must be set to *N* before the data aggregation is completed for the forecast run type from the Setup train stop in the Manage Forecast Configurations screen.

Forecast Method: Life Cycle

This forecast method requires data from INVENTORY.csv. In the presence of inventory data in the env, the PMO_AGGR_INVENTORY_DATA_FLG parameter value in RSE_CONFIG table (APPL_CODE = PMO) available in the Manage System Configurations screen must be set to *Y* before the data aggregation is completed for the forecast run type from the Setup train stop in the Manage Forecast Configurations screen. The requirement is to provide the inventory data for all active items, as well as on an ongoing basis. It is recommended to provide the inventory data for all items, at least for the most recent two quarters. The inventory information is used to calculate the base demand, using Bayesian methods, of the items and also to obtain store counts for active items.

For items with extremely low ROS, non-Bayesian methods will over-forecast. So, an inventory *informed* forecast (Bayesian methods) is determined. This is different from an inventory *constrained* forecast. A demand forecast, and not sales forecast, is determined. For example, 15 units might be forecasted, even if inventory is 10 units, so it is not bounded by inventory. The role of inventory is to be used as a prior hint in the Bayesian methods, so that 100 units are not forecasted for an item that has an initial inventory of 10.

Forecast Method: Automatic Exponential Smoothing

This forecast method does not require data from INVENTORY.csv. In the absence of inventory data from the env, the PMO_AGGR_INVENTORY_DATA_FLG parameter value in RSE_CONFIG table (APPL_CODE = PMO) available in the Manage System Configurations screen must be set to *N* before the data aggregation is done for the forecast run type from the Setup train stop in the Manage Forecast Configurations screen.

Forecast Method: Hierarchical Bayesian

This forecast method may require data from INVENTORY.csv. It supports inventory units as a feature, and thus, if there is a desire to consider the impacts of the inventory on the forecast,

then that data must be provided for about 18 months, and also on an ongoing basis. The PMO_AGGR_INVENTORY_DATA_FLG parameter value in RSE_CONFIG table (APPL_CODE = PMO) available in the Manage System Configurations screen must be set up accordingly before the data aggregation is done for the forecast run type from the Setup train stop in the Manage Forecast Configurations screen.

PRICE.csv

Forecast Method: Sales & Promo

This forecast method may require data from PRICE.csv. It supports price as a feature in the promotion module, and thus, if there is a need to consider the impacts of price changes on the forecast, then that data must be provided. The recommendation is to provide price history proportional to the length of the sales history provided, as well as on an ongoing basis. Note the following:

1. This method uses the latest price change as of the week ending, so it is not necessary to send daily price changes.
2. PRICE_CHANGE_TRAN_TYPE=0 is used to identify the full price or original price of the item. More than one record can have a value of 0 in situations such as if an item is repeated, or reset, or when the original or full price changes. This record is required for every item/location that had sale at any point in the history.
3. PRICE_CHANGE_TRAN_TYPE=4 records are used to identify the regular price changes for the item. These records help identify regular price changes and thus calculate the impact of the regular price change.
4. PRICE_CHANGE_TRAN_TYPE=8 records are used to identify the ticket price changes for the item. These records help identify markdowns and calculate the price impact for a markdown. If these records are not provided, the system will use a fallback option to calculate the price response, but it might result in lower forecast accuracies.

Forecast Method: Life Cycle

This forecast method may require data from PRICE.csv. Price data is used to identify the appropriate price changes to calculate the most accurate price response behavior. In the absence of this data, the sales data used in base demand calculation will not be de-priced. The recommendation is to provide price history proportional to the length of the sales history provided, as well as on an ongoing basis. Note the following:

1. This method uses the latest price change as of the week ending, so it is not necessary to send daily price changes.
2. PRICE_CHANGE_TRAN_TYPE=0 is used to identify the full price or original price of the item. More than one record can have a value of 0 in situations such as if an item is repeated, or reset, or when the original or full price changes. This record is required for every item/location that had a sale at any point in the history.
3. PRICE_CHANGE_TRAN_TYPE=4 records are used to identify the regular price changes for the item. These records help identify regular price changes and thus calculate the impact of the regular price change.
4. PRICE_CHANGE_TRAN_TYPE=8 records are used to identify the ticket price changes for the item. These records help identify markdowns and calculate the price impact for a markdown. If these records are not provided, the system will use a fallback option to calculate the price response, but it might result in lower forecast accuracies.

Forecast Method: Automatic Exponential Smoothing

This forecast method does not require data from PRICE.csv.

Forecast Method: Hierarchical Bayesian

This forecast method requires data from PRICE.csv. Price data is used to identify the appropriate price changes to calculate the most accurate price response behavior. The requirement is to provide price history proportional to the length of the sales history provided, as well as on an ongoing basis. Note the following:

1. This method uses the latest price change as of the week ending, so it is not necessary to send daily price changes.
2. PRICE_CHANGE_TRAN_TYPE=0 is used to identify the full price or original price of the item. More than one record can have a value of 0 in situations such as if an item is repeated, or reset, or when the original or full price changes. This record is required for every item/location that had sale at any point in the history.
3. PRICE_CHANGE_TRAN_TYPE=4 records are used to identify the regular price changes for the item. These records help identify regular price changes and thus calculate the impact of the regular price change
4. PRICE_CHANGE_TRAN_TYPE=8 records are used to identify the ticket price changes for the item. These records help identify markdowns and calculate the price impact for a markdown. If these records are not provided, the system will use a fallback option to calculate the price response, but it might result in lower forecast accuracies.

Data Correction and Reload Process

As part of building the dataset and reviewing the ETL Process, it is important to understand what is possible for correcting data after you have loaded it, as this will impact the amount of time you spend validating the data up-front before attempting to send it to RAP solutions.

SALES.csv and RECEIPT.csv

Sales and receipts are considered transactional interfaces, meaning that they load the data as incremental deltas. If a record is new, then it will be inserted into the database, but if it already exists then the incoming value will be SUMMED with the existing value. This summation is why deltas are used. The incoming data must only be the difference between the existing and expected values.

When you need to reload sales or receipt history due to data issues, or you need to revise existing data loaded in prior batch cycles, you have three options:

1. It is ideal to calculate the adjustments required and load those delta values on top of the current data to apply the corrections. This does not require any cleanup or resetting of the existing data. This is how revisions must be done during nightly batches.
2. Delete the existing sales or receipts data and reload it cleanly with the complete dataset that has been corrected for any issues. This is a complete deletion of the database tables for all history. This is done during history loads to make major changes. You do not normally do this after go-live.
3. Raise an SR requesting that specific subsets of data be deleted so that you can only reload those records. The subset of data to be deleted must be defined by a grouping of product, location, or calendar hierarchies; it must not be made up of arbitrary records. For example, delete all data for department 1001 so that the history for it can be re-loaded. Oracle will delete the data and then you must load the corrected data using the standard implementation processes.

When you are posting updates to existing records (option 1 above), you must consider the following:

1. For all of the dimension values, including item, loc, date, transaction ID, promo ID, retail type, and so on, the existing record must match the new revision so that the correct row is updated. If any one value does not match, then the revision may get inserted as a new row instead.
2. The value you are modifying must be the difference between the old and new value, while values you are not modifying must be provided as zeros. For example, updating SLS_QTY from 3 to 4 must have SLS_QTY=1 on the incoming revision and 0 on other columns.
3. For sales only, REVISION_NUM must be 2 or greater if you want to keep track of the revision count; however, it is not required. Receipts and other transaction types do not have revision number tracking.
4. Regardless of the data levels (aggregated item/loc/week or daily transactions), the correction process is the same. The only difference is how you configured the columns such as SLS_TRX_ID or PROMO_ID to work, because you must use the same values each time you post revisions.

Note that for receipt corrections for history data before daily batches, you can reload data incrementally to change the numbers such as INVRC_QTY, but if you are adding or removing entire receipt transactions then you must delete and reload both receipts and inventory in parallel as defined in the RAP implementation steps. This assumes you require the first/last receipt date calculations, as discussed previously. The receipts are applied to inventory calculations during the inventory history load, not during the receipts history load, so they must happen in parallel. If you do not require first/last receipt date calculations, then the receipts can be loaded separately from inventory.

INVENTORY.csv

Inventory is a positional interface; that is, it tracks the current positions of your inventory levels at each point in time. Positional values cannot be updated or overwritten once loaded. It is assumed that you provided the exact value of the position for each item/loc/date. It does not make sense to adjust those values later. For example, if your on-hand inventory units were five today, then that was the physical count of units you had at that time. Any changes that occur later only apply to some future date, not to past dates.

When you must reload inventory history due to data issues, you have two options:

1. Delete the existing inventory data and reload it cleanly with the complete dataset that has been corrected for any issues. This is a complete deletion of the database tables for all history. This is done during history loads to make major changes; you would not normally do this after go-live. As noted in the receipts sections, if you delete and reload inventory you want to do the same for receipts, assuming you require receipt date calculations for forecasting.
2. Raise an SR requesting that specific subsets of data be deleted so that you can only reload those records. The subset of data to be deleted must be defined by a grouping of product, location, or calendar hierarchies; it must not be a group of arbitrary records. For example, delete all data for department 1001 so the history for it can be reloaded. Oracle will delete the data and then you must load the corrected data using the standard implementation processes. Request the same deletion for receipts data and reload both in parallel if you are using receipts for forecasting.

When reloading inventory, if first/last receipt dates are required, then you must also provide receipts data for the same item/loc/weeks in the RECEIPT.csv at the same time, otherwise you will lose the receipt date calculations in your reloaded inventory data.

Once you have started nightly batches, you cannot back-post inventory data; your inventory updates will always be posted for the current business date in RAP. If inventory data was not posted for 1+ prior days for any reason, such as issues in your source system or problems at

the stores, then include those updates in the next batch and they will be posted for the current date, setting the inventory position values to be correct from that day forward.

PRICE.csv

Price data is a positional interface similar to inventory, but it has additional complexities that make it even more challenging to modify once loaded. Price data calculates many metrics over time during the load process, such as the number of markdowns that have occurred or the last known clearance price. These values are dynamic; they are updated every day as price changes in the history files are processed in chronological order. Once the data is loaded, you cannot directly alter or update it. Your options for data corrections are:

1. Delete the existing price data and reload it cleanly with the complete dataset that has been corrected for any issues.
2. Raise an SR requesting that specific subsets of data be deleted so that you can only reload those records. The subset of data to be deleted must be defined by a grouping of product or location hierarchies; it must not be arbitrary records and it cannot be a subset of calendar periods. Oracle must delete all data for the entire calendar range so derived metrics can be recalculated by any new loads. For example, delete all data for department 1001 so Oracle can reload the history for it. Oracle will delete the data and then you must load the corrected data using the standard implementation processes.

AIF Questions

Here are some answers to typical questions related to AIF.

What is the flow of sales data in AIF that comes from RI?

Sales data from RI first comes into RSE_SLS_TXN (transaction level) and then aggregated data flows into RSE_SLS_PR_LC_DY (sku/store/day), RSE_SLS_PR_LC_WK (sku/store/week), and RSE_SLS_PR_LC_CS_WK (sku/store/custseg/week).

Which table stores the aggregated sales data for forecasting at various intersection levels?

PMO_ACTIVITIES (main table for all forecast methods) and PMO_CUM_SLS (cumulative sales table only used by the Life Cycle and Hierarchical Bayesian forecast methods) stores aggregated weekly sales data for each unique AGGR_LVL_KEY. These tables obtain data from RSE_SLS_PR_LC_CS_WK.

Why is the newly loaded data not reflected in the forecast results?

Validate that the sales data is present in the below tables. The tables are populated in the following sequence:

rse_sls_txn --> rse_sls_pr_lc_wk --> rse_sls_pr_lc_cs_wk --> pmo_activities

Refer to the Configuration and Main Data Load Jobs table for the POM jobs that populate the above tables.

Also validate the last processed calendar hierarchy id using the following query:

```
select * from rse_proc_stat where PROC_NAME like 'PMO_%' or PROC_NAME like 'RSE_SLS_%'
```

Validate that the LAST_PROCESSED_HIER_ID for the weekly tables and pmo_activities is the same.

Why is the run type that I created not showing up in the Test screen?

Aggregation must be completed for run types to show up in the Test screen to be usable to do forecast runs for those run types. For a run type based on Life Cycle or Hierarchical Bayesian forecast method, Cumulative aggregation must also be completed.

How much data do I require for forecasting?

Data requirements differ by forecast methods. The default method for IPO is Sales & Promo forecast method and it requires at least 104 weeks of sales history at each of the different AGGR_LVL_KEY to be used by this method. This can be verified from the PMO_ACTIVITIES table filtering on the desired AGGR_LVL_KEY. prod/locs that have at least 104 weeks between their first and last sales will have a seasonality curve generated. So, if in total, there is at least 104 weeks of sales history available but none of the prod/locs at an AGGR_LVL_KEY satisfies the condition then this method may not work at that AGGR_LVL_KEY. Other forecast methods such as Life Cycle will require inventory and receipts data, in addition. Refer to [Data Requirements for Various Forecast Methods](#) for more information.

What is the difference between forecast run type and run?

A forecast run type can be thought of as a collection of estimation and forecast runs grouped by some important parameters such as hierarchy type (merchandise/location), forecast intersection (merchandise/location) level, forecast method, forecast measure, data source, etc. All estimation and forecast runs within a particular run type will have the same values for these important parameters. An estimation or forecast run cannot be part of more than one run type. Similarly, a forecast run type always belongs to a unique AGGR_LVL_KEY, but an AGGR_LVL_KEY can have multiple run types mapped to it. The AGGR_LVL_KEY column in the table RSE_FCST_RUN_TYPE shows the AGGR_LVL_KEY a particular run type is based on, and the table RSE_AGGR_LVL_KEY shows all the parameters an AGGR_LVL_KEY is based on.

What is the difference between forecast run and estimation run?

Any forecast run and estimation run belongs to a unique run type. Estimation run denotes the pre-processing and parameter estimation stages. Forecast run denotes the base demand calculation and forecast generation stages. The same estimation run may be used by multiple forecast runs but each forecast run is dependent on a unique estimation run.

What is the use of DFLT_LIFE_CYCLE parameter?

DFLT_LIFE_CYCLE is a parameter in the RSE_CONFIG table (APPL_CODE = RSE) and can be accessed from the Manage System Configurations screen. It works together with the data in RSE_FCST_LIFE_CYCLE_CLSF table. As a default configuration, RSE_FCST_LIFE_CYCLE_CLSF table should be empty. DFLT_LIFE_CYCLE should be set to LLC when using the Sales & Promo forecast method, and to SLC when using the Life Cycle forecast method. Other forecast methods are not impacted by DFLT_LIFE_CYCLE. Also, forecast run types based on the Life Cycle forecast method and mapped to LPO - Promotion/Markdown will treat the value of the DFLT_LIFE_CYCLE as SLC.

Can I map one forecast run type to more than one application?

Yes, a forecast run type can be mapped to multiple applications. Each run type-application mapping has a specific usage. While setting up a mapping with IPO-DF application, an External Run type Key must be provided which is used by the IPO-DF UI to represent this run type.

What is the source for approved forecast?

Approved forecast source is determined for each run type independently. By default, it is AIF unless the run type is mapped to IPO-DF application in which case it is RDX. This holds even if the run type is mapped to other apps in addition to IPO-DF.

At what level must IPO-DF send pre-forecast batch parameters to AIF?

AIF requires all the pre-forecast batch parameters at each of the different forecast levels that IPO-DF user wants to generate forecast. In other words, AIF requires these parameters for each of the different run types mapped to IPO-DF.

At what level does AIF require Out-of-Stock and Outlier indicators?

Out-of-Stock and Outlier indicators are used by the Sales & Promo forecast method during the pre-processing steps. Pre-processing happens at the forecast intersection level. So, these indicators are needed at each of the different forecast intersection levels for the Sales & Promo method.

Why are the IPO-DF forecast results in AIF not getting exported to RDX?

Export of IPO-DF forecast results from AIF to RDX depends on 2 factors: a particular prod/loc has a forecast generated and that prod/loc is present in the table RSE_RDF_FCST_RUN_TYPE_PRM for the relevant run type. Note that this table is populated from the RDX table RDF_FCST_PARM_EXP during the pre-forecast batch. So, one must determine if there are any common prod/loc between RSE_RDF_FCST_RUN_TYPE_PRM filtered by the relevant run type and RSE_LLC_FCST_BLINE (for Sales & Promo method) or RSE_DMD_FCST (for Life Cycle method) filtered by the relevant forecast run. If there are no common prod/loc, then no export will happen from AIF to RDX. Either change the prod/loc sent from IPO-DF during the pre-forecast batch or update the data files so that the desired prod/loc gets forecast generated in AIF.

What checks must I make to ensure that AIF-IPO-DF workflow is ready to go?

The desired forecast run type must be mapped to IPO-DF and assigned an External Run type Key. The IPO-DF user must send pre-forecast batch parameters using the same External Run type Key. The run type should be in Active state. The completed forecast run that must be exported from AIF to RDX must be Approved.

Why am I not able to export out a particular forecast run from AIF to RDX?

A forecast run can only be exported out once. After that the run is considered dead and cannot be exported again. It can be verified from the table RSE_FCST_RUN_HDR filtered by the ID column which is the forecast run ID and see the EXPORTED_DT column value. This remains empty until the forecast run is exported and gets populated when the export happens. Once this value gets populated, the same forecast run cannot be exported again.

Can I use a different forecast method for MFP and AP other than Auto Exponential Smoothing?

Yes, Sales & Promo method and Life Cycle method are also supported for MFP and AP. Once a run type is setup based on Sales & Promo method or Life Cycle method and mapped to MFP or AP, the forecast exports from that run type will go to the RDX table RSE_FCST_DEMAND_EXP which is where MFP and AP reads the forecasts from.

How to check the seasonality curves used in forecasting?

RSE_SEAS_CURVE_DIM stores the reliable seasonality curves for each estimation run. ROW_PROC_LEVEL of 1 represents the escalation level at which each seasonality curve got computed. ROW_PROC_LEVEL of 3 represents the forecast level. So, if the forecast level is part of the escalation path, and if a curve that got computed for a forecast level prod/loc is not pruned then the same prod/loc will appear for that curve for both ROW_PROC_LEVEL 1 and 3. RSE_SEAS_CURVE_WK stores the seasonality indices for each estimation run and each curve.

How to check the seasonality curves that are pruned and why those got pruned?

For each estimation run, PMO_RUN_LLC_SEASONALITY_CURVE (for Sales & Promo method) and PMO_RUN_SEASONALITY_CURVE (for Life Cycle method) stores all the curves (both pruned and reliable) generated along with the reason of pruning.

Why am I not able to see the approved forecast in RI and/or RMS?

The relevant forecast run type must be mapped to the RI/RMS application for the approved forecast to flow to RI for BI reporting and/or RMS for merchandising. If the RI/RMS application is not visible in the Map screen drop-down list, then it can be turned on from the RSE_APP_SOURCE table in Manage System Configurations screen.

How to turn off the Outlier calculation GA feature to be able to implement a custom solution?

If the GA feature for Outlier indicator calculation is not required, then it can be turned off by setting PMO_OUTLIER_CALCULATION_FLG to N from the RSE_CONFIG table (APPL_CODE = PMO) in Manage System Configurations screen.

How can I use alternate hierarchy in AIF for forecasting?

The desired alternate hierarchy must first be constructed in AIF. Refer to [Building an Alternate Hierarchy in AIF Applications](#) for more information. The merchandise and location alternate hierarchies must then be set up correctly in the table RSE_FCST_DFLT_RUN_TYPE_PARAM via the Manage System Configurations screen. Finally, those alternate hierarchies need to be used while creating the run types.

How can I set up custom processes for forecasting through IW?

Different custom processes are required for different stages of the forecasting pipeline. So, to execute such processes during weekly batches through POM, the correct custom POM jobs must be used which will ensure the desired POM dependencies on pre and post POM processes. Refer to [Custom Jobs Through Innovation Workbench \(IW\)](#) for more information.

How can I set up Fact Measures data for forecasting in AIF?

Different configuration steps must be followed in AIF to consume the aggregated sales data coming through the RI fact interfaces. Refer to the [Configurations in AIF to Generate Forecasts for FACT Measures](#) for more information. Automatic Exponential Smoothing forecast method can use the fact measures data source by creating the desired run type correctly.

Why did the additional week of data loaded into AIF did not get aggregated for some AGGR_LVL_KEY?

For the additional week of data to get aggregated in PMO_ACTIVITIES for a particular AGGR_LVL_KEY, at least one of the corresponding run types based on that AGGR_LVL_KEY

must be in Active state. Run the relevant jobs in PMO_ACTIVITY_LOAD_ADHOC_PROCESS with the necessary parameters to sync up the data.

How can I check the DB logs for errors for the different processes running in AIF?

DB logs are stored in RSE_LOG_MSG. To see the latest logs, ORDER BY MSG_TS DESC. To check for errors, filter by LOG_LEVEL = 'ERROR'. To see the progress of a particular process, filter by PROGRAM_UNIT or ROUTINE_NAME. The PROCESS_ID for some logs serve as the estimation run ID or the forecast run ID, but one must be careful with its usage as the PROCESS_ID might also mean something else. The AUDSID remains the same for a particular process execution so that can also be used as a filter to check the logs for a particular process.

Why are the forecasting estimation runs completing in a few seconds and don't seem to do anything?

For the submitted estimation runs to execute the different stages properly, there has to be seed data present in PMO_RUN_STAGE_EXEC_REQUEST for RUN_HDR_ID of -1. Also, data must be present in both PMO_RUN_STAGE_EXEC_REQUEST and PMO_RUN_STAGE_EXECUTION for the relevant estimation runs.

Why did the base demand fail or complete with zero rows for the Sales & Promo forecast method?

One of the primary reasons could be no seasonality data in RSE_SEAS_CURVE_DIM for the relevant estimation run. This, in turn, could be because of the DFLT_LIFE_CYCLE parameter not set to LLC or due to insufficient sales history available in PMO_ACTIVITIES for the relevant AGGR_LVL_KEY.

Why did the base demand fail or complete with zero rows for the Life Cycle forecast method?

One of the primary reasons could be no seasonality data in RSE_SEAS_CURVE_DIM for the relevant estimation run or no in-season base demand data in RSE_BD_IN_SEAS_SLS for the relevant forecast run. This, in turn, could be because of the DFLT_LIFE_CYCLE parameter not set to SLC or due to insufficient inventory data and old first receipt dates.

Why did the forecast run not generate the new item forecast?

New item data should be populated in RSE_FCST_NEW_ITEM_PARAM for the relevant forecast run type before the forecast run is executed. The new item base demand is copied from the like item for the same location. So, the like item must have base demand generated for that particular location.

Why is the RUN_ID not same for RSE_FCST_RUN_TYPE_CONFIG and RSE_FCST_RUN_HDR_EXP?

The RUN_ID in RSE_FCST_RUN_TYPE_CONFIG and RSE_FCST_RUN_HDR_EXP are different by design as these two RDX tables are populated by separate processes. The RUN_ID in RDX tables is assigned incrementally every time a new process writes something to one of the RDX tables. Also, the RUN_ID in RDX tables is totally different than the forecast or estimation run IDs in AIF.

Can I forecast seasonal items with Sales & Promo method or long-running items with Life Cycle method?

Both the Sales & Promo and the Life Cycle methods are capable of forecasting all kinds of items. The choice of the forecast method depends on multiple factors such as data availability, business model of the retailer, purpose of the forecast output, and so on. The default method

for IPO customers should be the Sales & Promo method. They should only switch to other methods with proper discussions and understanding of their requirements.

Inventory Planning Optimization

This chapter describes the Inventory Planning Optimization (IPO).

Overview

Inventory Planning Optimization (IPO) determines the optimal time-phased replenishment plan which consists of the replenishment policies, that is the reorder point (RP) and receive up-to level (RUTL), and the recommended purchase order (PO), transfer and allocations at item/location/day level for the configurable planning horizon. The optimal plan is generated using simulation and optimization methods and considers inputs like supply chain network, replenishment attributes and business rules. Moreover, the optimization engine uses machine learning methods and simulation-based optimization to calculate the trade-offs between the service level and the inventory cost for different replenishment policies. The trade-off analysis is leveraged to generate the optimal replenishment policies for achieving a desired target service level. In addition to the replenishment plan, IPO-IO recommends optimal rebalancing transfers between stores to increase sell-through and to avoid markdowns. This type of recommendation can be turned off when not applicable (for example, for grocery categories).

The data-driven replenishment policies, PO/transfers, and rebalancing transfers are pushed to Oracle Retail Merchandising System (RMS) to execute purchase orders and transfers. Optionally, the retailer can choose to send only the replenishment policies to RMS and have the final order quantity and PO/transfers be calculated and generated in RMS.

IPO leverages historical sales, inventory positions, replenishment attributes such as lead time and review schedule, business requirements such as store priorities for shortfall reconciliation, and the demand forecast to generate the optimal time-phased replenishment plan. The demand forecast that is generated by the forecast engine within AI Foundation considers different factors such as price effect, holidays, and promotions, and variation across customer segments.

Inventory Planning Optimization Runs

Types of Runs

IPO supports two kinds of runs: user runs and batch runs. Batch runs are scheduled to run automatically at regular intervals (for example, weekly or daily). By default, the batch runs are scheduled for daily execution. User runs are typically created by the user to do what-if analysis and/or to override the recommendations generated by the latest batch run. IPO runs generate recommendations at the item-location level. The items that are included in a run can be selected by specifying one, a multiple, or all of the nodes at a higher merchandise level, such as department. This level is configurable. Similarly, the locations that are to be included in a run can be selected by specifying one, a multiple, or all of the nodes at a higher location level, such as area. This level is configurable.

Each run will generate one or multiple type of recommendations depending on the type of analysis that are selected when creating the run from UI. For batch runs, the type of analysis is controlled by the following configurations in RSE_CONFIG.

IO_BATCH_RUN_TIME_PHASED_FLG: Y/N flag that indicates whether time phased simulation is performed as part of batch run. The default value is Y.

IO_BATCH_RUN_INV_REBAL_FLG: Y/N flag that indicates whether inventory rebalancing analysis is performed as part of batch run. The default value is N.

IO_TIME_PHASED_OPT_PO_TSF_FLG: Y/N flag to indicate whether the time phased optimization should generate the PO/Tsf recommendations. Default value is Y. (When the value is N, only the replenishment policies is generated.)

Time-Phased Planning

Time-phased planning is the main type of analysis that must run as part of the batch, so the replenishment policies and PO/transfers are generated daily. If the implementation scope is to generate replenishment policies only, set **IO_TIME_PHASED_OPT_PO_TSF_FLG** to N.

Inventory Rebalancing

Inventory rebalancing can be enabled in as part of the batch if it's relevant (for example for fashion retailers) and if there is a need to generate rebalancing recommendations daily. Alternatively, the rebalancing runs can be done as-hoc from UI when needed.

Trade-Off Analysis

The trade-off analysis is not a kind of process that needs to be run daily because the changes in sales patterns, replenishment rules and supply chain network are not that frequent. Therefore, the optimal trade-off curves are typically recommended to be updated quarterly or when there is a major change in one of the factors mentioned above.

Note

It is strongly recommended not to enable the trade-off analysis for the batch runs. This can make the batch runtime long since the trade-off analysis performs extensive data processing and large-scale simulation.

Inventory Planning Optimization UI Workflow

The IPO UI workflow consists of the following:

- Manage recommendations:
 - Replenishment: Provides a calendar view of different recommendations and a separate view for each type of recommendations: Replenishment Policy, Rebalancing Transfers, PO and Transfers.
 - Allocation: Provides a view of initial and cross-dock allocations and details of each allocation.
 - Inventory Plan: Provides a time-phased view of different inventory plan metrics and alerts. Also, allows user to change strategy and perform what-if analysis.
- Overview:
 - Run Overview: Provides summary of inputs and outputs of batch and ad-hoc runs. user can create a run, copy a run, open a run, or delete a run. When the user clicks on a successful run, it is opened in a new tab with three sub-tabs to show replenishment policies, rebalancing transfers, and PO and transfers, as well as a sub-tab to show the summary of inputs of the run. For a failed run, only the input summary sub-tab is

displayed. To create a run, the user must specify the run scope and strategy. As part of the run scope, the user specifies the areas and departments for the run and the type of analysis to be performed. As part of run strategy, the user can specify the business rules.

- Create Run: User can create ad-hoc runs and specify the run scope and the strategy that will be used to determine the business rules for the run.
- Forecast Overview: Provides a view of forecast at SKU/location/customer segment level.
- Inventory Overview: provides a view of the stock levels and the historical trend of the inventory at the item-location level.

The following sections explain the main implementation tasks that involve configuring the following:

- The roles and permissions assigned to users.
- The loading of retailer data.
- The configuration parameters.
- The business rules that determine constraints that the application takes into account during the optimization process.

Security

User roles are used to set up application user accounts through Oracle Identity Management (OIM). See *Oracle Retail AI Foundation Cloud Services Administration Guide* for details. The following roles are required.

- ADMINISTRATOR_JOB in order to access the Control & Tactical Center
- INVENTORY_ANALYST_JOB in order to access the IPO application.

In addition to above roles, verify the following:

- Access to Innovation Workbench and/or Data Visualizer: This is necessary in order to query or visualize the data and verify that the data loaded matches the desired expectations.
- Access to POM to execute ad-hoc and batch jobs: The POM UI URL is something like <host>/POMJetUI. If the user cannot access the POM UI, contact the administrator to obtain the relevant access/user roles.

Data Load Requirements

This section provides information about setting up the data that the IPO application uses, including guidelines regarding the expectations for the data element requested and where it is used. Information about these files can be found in *Oracle Retail Insights Cloud Service Suite/ Oracle Retail Analytics and Planning Cloud Services Data Interface*.

Most of the AI Foundation Cloud Services (AIF) data is pushed in two-step process. Any additional IPO-specific data is directly pushed into the AIF.

1. First, data is loaded, using CSV and W_ interfaces and jobs in the AIF DATA standalone schedule in POM. Then, RADM_REFRESH_JOB must be run to refresh the table statistics before any AIFApps job is run (though REFRESH_RADM_JOB is automatically executed as part of most ad hoc processes in AIF Data).

2. Second, appropriate AIF applications jobs are used to push data into AIF applications. At the time of implementation, the user will only use ADHOC jobs, not any batch jobs. Batch jobs (described in "POM Jobs") are necessary to put the system on a batch schedule.

Note that some of the jobs require relevant configuration parameters to be specified with client-specific values. If incorrect configuration values are used, a job may run without errors, but will not produce the desired data in the target tables. If the client or implementation team wants to view the progress of the job or any errors in order to file an SR, then AIF provides database logging in a table called RSE_LOG_MSG. In addition, analytic logs generated by the programs that run optimization can be found in RSE_ANALYTIC_LOG_MSG. More details are provided in the Troubleshooting section in this chapter.

RAP Foundation Data

These interfaces provide the foundation data for the AI Foundation Cloud Services. First, the user must execute and verify that all the RAP foundation data has been loaded.

For IPO, the following data is required:

- Product Hierarchy (PRODUCT.csv). When Style, Style Color must be loaded, appropriate columns must be populated.
- Location Hierarchy (ORGANIZATION.csv)
- Sales. It is recommended to load at least 24 months of historical data to obtain a good signal in the model training and a reliable parameter estimation for the demand forecasting.
- Inventory. (INVENTORY.csv). For the time-phased planning, it is sufficient to have the latest inventory position (i.e., quantity of on-hand, on-order, in-transit, etc. for the latest day). For trade-off analysis, it is recommended to provide historical inventory which will be used for lost sales correction before performing simulation. If not provided, the trade-off analysis will be performed on the sales time series. While this is not ideal, for most categories the final trade-off curves are not too sensitive to the lost sales correction when the KPIs are measured at aggregate level.
- Inventory Receipts. (RECEIPT.csv). This data is used to identify the first and last receipt at item/location level. It is used by IPO to update the life cycle start date, which is being used to determine the life stage of the item/location based on the life cycle rules (for more details please refer to the Rules & Strategies chapter and the IPO rules). So, the inventory receipt is optional in IPO depending on what kind of rules are being set up and used. From forecasting standpoint, the inventory receipt must be provided for forecasting fashion items that are considered short life cycle.
- Customer Segments. Even when the retailer is not planning to load or use customer segments, a dummy record is required. The AIF ad hoc jobs will create a dummy record when the source data is not provided, and it is not required for the client or implementer to provide these interfaces if they are not going to use them: W_PARTY_PER_DS and W_RTL_CUSTSEG_DS.
- Price and Cost (PRICE.csv). This data is required for calculating some of the KPIs that are shown in IPO-IO screens. Moreover, price data must be provided for forecasting fashion items that are considered short life cycle.
- Product Images. The URLs for rendering product images in IPO-IO screens can be provided.
- Promotions (PROMOTION.csv). This is an optional data for IPO-IO. It is used in showing the promotion flag at item/location/day level in some of the UI charts. Moreover, promotions must be provided for forecasting fashion items that are considered short life cycle; otherwise, the forecast will not reflect the promotion effects and will be inaccurate.

- Replenishment attributes (PROD_LOC_REPL.csv). This data is required for all type of runs and analysis in IPO-IO. The required attributes include but not limited to primary supplier/warehouse source, lead time and replenishment method. For more details, please refer to the RAP interface guide.
- Review schedule (W_RTL_REPL_DAY_DS.dat). This data is required for all type of runs and analysis in IPO-IO.
- Supplier (SUPPLIER.CSV). This data is required for all type of runs and analysis in IPO-IO.

The following data is optional and can be provided depending on the business use cases.

- Distribution networks (REPL_DISTRO.csv)
- Internal review cycles (REPL_REV_INT.csv)
- Supplier review cycles (REPL_REV_SUP.csv)
- Internal lead times (REPL_LT_INT.csv)
- Supplier lead times (REPL_LT_SUP.csv)
- Procurement networks (REPL_PROC.csv)
- Internal order multiples (REPL_MULT_INT.csv)
- Supplier order multiples (REPL_MULT_SUP.csv)
- Rounding levels (REPL_ROUND.csv)
- Scaling rules (W_RTL_REPL_SUP_DIM_DS.dat)

For example, if there is a need to define time-dimensioned replenishment attributes (e.g. longer lead time during holidays and shorter lead time at other times of the year), it will have to be provided using the REPL_LT_INT.csv and REPL_LT_SUP.csv. Another example is defining a warehouse-to-warehouse relationship in the supply chain network. For customers who use RMS, it will not be possible to have a source warehouse defined for a warehouse location. If there is a need to define such a link in the supply chain network, it will have to be provided using REPL_DISTRO.csv.

Before pushing any data into the AIF, it is critical to make sure RAP foundation data is accurate and verified. Verification can be done through Data Visualizer or through the Innovation Workbench. Re-loading some of the RAP foundation data elements is non-trivial as it will require an SR to reset/truncate some of the target tables.

RSP Foundation Data (RSE_% Tables)

Once the interface configuration is complete and the RAP Foundation data has been verified, the user can push the data from RAP into AIF. As mentioned before, once database logging has been enabled, users can check for any errors and the progress of the process in RSE_LOG_MSG. Before running the RSE_MASTER_ADHOC with appropriate flags to execute the following jobs, the user must set the correct values for configuration parameters in RSE_CONFIG. This can be done using Manage System Configurations (as described in Control and Tactical Center) and then selecting RSE_CONFIG from the drop-down list.

Inventory Planning Optimization Data (IO_% Tables)

Once RSE_MASTER_ADHOC is successfully run, data can be loaded to IPO tables by running the IO_MASTER_ADHOC.

Data Input Requirements

This section provides information about setting up the data that the IPO application uses, including guidelines on the expectations for the data element requested and where it is used. Information about these files can be found in Oracle Retail Insights Cloud Service Suite/Oracle Retail Analytics and Planning Cloud Services Data Interface.

Hierarchy Data

Hierarchies are part of the core data elements that are used in IPO, in both the forecasting and optimization modules. The four types of hierarchies are Location Hierarchy, Merchandise Hierarchy, Calendar Hierarchy, and Customer Segments Hierarchy. Most of the interfaces have numerous columns and thus, it is possible to define which columns will be populated in the interface by specifying the column headers (the order of columns does not have to be the same) in the corresponding file, such as PRODUCT.csv.

Location Hierarchy

Here is an example of a location hierarchy: CHAIN ' COUNTRY ' REGION ' DISTRICT ' STORE. The run's scope level must match a node in the location hierarchy. For example, you can specify the run scope at the Area level and include one, multiple, or all areas in the run. Batch runs by default include all values of the scope level (for example, all areas). Note that the E-com channel can be defined as part of the location hierarchy. Relevant interfaces are: W_INT_ORG_ATTR_D, W_INT_ORG_DH, W_INT_ORG_D. These are all loaded using the ORGANIZATION.csv file.

Merchandise Hierarchy

An example of merchandise hierarchy is as follows: CHAIN 'COMPANY/BANNER 'DIVISION ' DEPARTMENT 'CLASS ' SUBCLASS ' STYLE 'COLOR ' SIZE (SKU). There can be up to nine levels in the merchandise hierarchy. When IPO-IO is used for fashion apparel, it is expected that Style and Color be provided through the relevant interfaces.

The STYLE, COLOR and SIZE SKUs are expected to be provided in the W_PRODUCT_D interface, while the levels between CHAIN and Subclass are provided via W_PROD_CAT_DH interface. In addition, there must be consistency among the levels provided when using an extended hierarchy. W_PRODUCT_ATTR_D, is used to indicate the relationship between Style, Style/Color, and Style/Color/Size for an extended hierarchy. All of these tables will be populated when providing a PRODUCT.csv file as a foundation file input. If the retailer wants to use an extended merchandise hierarchy, then the retailer must populate this interface.

Here are the specific fields:

- PRODUCT_ATTR13_NAME = PROD_NUM for the Style (for example, 0000190086820900)
 - PRODUCT_ATTR14_NAME = PROD_NUM for the Style/Color (for example, 190086834203)
 - PRODUCT_ATTR15_NAME = PROD_NUM for the Style/Color/Size (for example, 1975699).
- The value of PROD_NUMs is the same as the value in the W_PRODUCT_DS.PROD_NUM interface.

Calendar Hierarchy

This is one of the core hierarchies. A retailer can specify the calendar depending on that retailer's business requirements (for example, fiscal calendar). The relevant interface for this is W_MCAL_PERIOD_D, loaded using CALENDAR.csv file.

Customer Segments Hierarchy

The AIF forecast engine in IPO supports demand forecasting at the customer segment level. This is leveraged by the optimization algorithm to recommend customer-aware rebalancing transfers. To use this functionality, the retailer must load customer segments. The relevant interfaces to supply this data are: W_RTL_CUSTSEG_DS and W_RTL_CUST_CUSTSEG_DS.

Runs in IPO-IO can be set up to run at configured levels for location and merchandise.

- **Location Level.** The location level of a run's scope is configurable in RSE_CONFIG, and it is denoted as IO_LOC_HIER_FILTER_LVL. For example, a typical level is Area.
- **Merchandise Level.** The merchandise level of a run's scope is configurable in RSE_CONFIG, and it is denoted as IO_PROD_HIER_FILTER_LVL. For example, a typical level is Department.

Retail Sales Data

Retail Sales data is a required interface and is provided through SALES.csv file (W_RTL_SLS_TRX_IT_LC_DY_FS staging table). IPO requires the retailer's historical sales data at the SKU-Store-Day level. When the system is in production, the latest incremental sales data is obtained as part of the batch process.

Inventory Data

Historical inventory data is a required at two levels: the historical weekly inventory at sku-location level and the inventory at sku-location for the latest day. The latest weekly inventory data is obtained as part of the weekly batch process. The daily inventory data for the latest day is obtained through the daily batch process. In both interfaces, various components of inventory, including on-hand, on-order, back-order, and so on, are required. Historical weekly inventory data as well as daily inventory data can be specified through INVENTORY.csv (staging table W_RTL_INV_IT_LC_DY_FS).

Product Attributes

Product attributes are not required for IPO, but they are useful for the demand forecasting, and can be used for demand forecasting of new items. Product attributes must use W_RTL_ITEM_GRP1_DS since it can support any number of attributes.

If the retailer wishes to load STYLE or COLOR as product attributes, then the W_RTL_ITEM_GRP1_DS interface is used to provide style and color attributes for the different products, using a value in PROD_GRP_TYPE of either STYLE or COLOR. The actual values for the Styles and Colors are provided in columns FLEX_ATTRIB_2_CHAR and FLEX_ATTRIB_4_CHAR using values that represent the ID for the Style (for example, 1234), and a description for the Style (for example, Loose Fit). The columns FLEX_ATTRIB_3_CHAR and FLEX_ATTRIB_10_CHAR contain the appropriate designation for STYLE or COLOR.

Product Images

The Images URL can be provided through W_RTL_PRODUCT_IMAGE_DS.dat interface. This interface has a column called PRODUCT_IMAGE_ADDR that contains the full URL to an image of the product. This URL must be in the following format: http[s]://servername[:port]/location/filename.extension

For example:

- PRODUCT_IMAGE_NAME = imagename.png
- PRODUCT_IMAGE_ADDR = http://hostname/url/imagename.png
- PRODUCT_IMAGE_DESC= Short description of the image

Replenishment Attributes

Replenishment attributes such as lead time, review time, presentation stock, and so on, are required for any sku-location that is expected to be processed by IPO to generate an optimal replenishment policy. These attributes must be provided at sku-location level through W_INVENTORY_PRODUCT_ATTR_DS.

Price and Cost

Price and cost data at the end of the week for each item-location is required. This data can be loaded with the PRICE.csv file through AIF Data jobs into the data warehouse, which places the data in the W_RTL_PRICE_IT_LC_DY table for AIF applications to consume. Alternatively, a retailer can directly load price and cost data to AIF using the W_RTL_PRICE_IT_LC_DY_FRI interface. Alternatively, a retailer can directly load price and cost data to AIF Apps using the RSE_PRICOST_PR_LC_WK_STG interface, which uses the rse_pricost_pr_lc_wk_stg.txt interface file..

Season

Data for the name of the season along with start and end dates can be provided through IO_SEASON_STG table (which is directly loaded to AIF applications) or SEASON.csv (which flows through the AIF data warehouse along with other foundation files). If the retailer does not have any seasons, then the retailer can define one long season that encompasses their entire calendar.

Global Configurations

[Table 15-1](#) shows the configurations that must be provided at the time of implementation. These configurations can be modified in RSE_CONFIG table using the Manage System Configuration screen under Control & Tactical Center in the AIF dashboard.

Table 15-1 Global Configuration Parameters

APPL CODE	Parameter Name	Parameter Value	Description
RSE	EXTENDED_HIERARCHY_S RC	Default value is NON-RMS.	Data source providing extended hierarchy data RMS/NON-RMS.
RSE	LOAD_EXTENDED_PROD_HIER	Default value is Y. If the product hierarchy data had 9 levels, keep this value as Y. If it has 7 levels, change this value to N.	Y/N Value. This parameter is used by the product hierarchy ETL to know if the extended product hierarchy is needed.

Table 15-1 (Cont.) Global Configuration Parameters

APPL CODE	Parameter Name	Parameter Value	Description
PMO	PMO_PROD_HIER_TYPE	Default value is 3. If the product hierarchy data has 9 levels (i.e. it has extended hierarchy), keep this value as 3. If it has 7 levels, change this value to 1.	The hierarchy ID to use for the product (Installation configuration).
PMO	PMO_AGGR_INVENTORY_D ATA_FLG	Default value is Y. Set this value to N if inventory data is not loaded (inventory data is not required for MFP forecasting but it is required for other applications like Lifecycle Pricing Optimization, Inventory Planning Optimization-Inventory Optimization and IPO-Demand Forecasting).	Specifies if inventory data is present and if it should be used when aggregating activities data.
RSE	PROD_HIER_SLSTXN_HIER _LEVEL_ID	Default value is 9.	This parameter identifies the hierarchy level at which sales transactions are provided (7-Style, 8-Style/color or 9 Style/color/Size). It MUST match the extended hierarchy leaf level.
IO	INV_REBALANCE_FLG	Y	The inventory rebalancing section in the UI create run screen will be shown or hidden based on this flag. Must be Y or N.
IO	IO_CURRENT_DATE	.	This value will be automatically updated to latest daily inventory date by ETL process. It should not be modified manually.
IO	IO_PROD_HIER_TYPE	Default value is 3. If the product hierarchy data has 9 levels (i.e., it has extended hierarchy), keep this value as 3. If it has 7 levels, change this value to 1.	Hierarchy Type Identifier for the Product/Merchandise Hierarchy Type.
IO	IO_REBAL_DEF_REGION	usastates	Specifies the default region in the rebalancing details map view
IO	IO_LOC_HIER_FILTER_LVL	3	Level Identifier for the level of the Location Hierarchy at which IPO-IO runs can be set up.

Table 15-1 (Cont.) Global Configuration Parameters

APPL CODE	Parameter Name	Parameter Value	Description
IO	IO_PROD_HIER_FILTER_LVL	4	Level Identifier for the level of the Product Hierarchy at which IPO-IO runs can be set up.
IO	IO_ALLOCATION_BUFFER_DAYS	7	The number of days before selling start date that the initial shipment must be received. This is used for initial allocation recommendations.
IO	IO_AUTO_APPR_NEW_PARAMS	N	Indicates whether net new parameters in a batch should be auto-approved. Default to N (No).
IO	IO_BATCH_RUN_INV_REBAL_FLG	N	Y/N flag that indicates whether inventory rebalancing analysis is performed as part of batch run.
IO	IO_BATCH_RUN_PO_TRANSF_R_RECOMM_FLG	Y	Y/N flag that indicates whether PO transfer recommendation is performed as part of batch run.
IO	IO_BATCH_RUN_TIME_PHASED_FLG	Y	Y/N flag that indicates whether time phased simulation is performed as part of batch run.
IO	IO_BATCH_RUN_TRADEOFF_SIM_FLG	N	Y/N flag that indicates whether trade-off simulation is performed as part of batch run.
IO	IO_CAL_HIER_PROCESSING_LVL	4	Level Identifier for the level of the Calendar Hierarchy at which we process Inventory. Default 4 is to process at Week Level.
IO	IO_TIME_PHASED_HORIZON	14	Number of periods in the future for which the time phased plan is generated.

Table 15-1 (Cont.) Global Configuration Parameters

APPL CODE	Parameter Name	Parameter Value	Description
IO	IO_ENABLE_NTIER_INTER FACE_FLG	N	Indicates source of data to populate IO_REPL_ATTR_PROD_LOC_EXP (Y,N,X). Possible values are: Y (data loaded from RI interfaces for replenishment attributes and review schedule (typically coming from RMS). In addition, data loaded into IO_FINAL_RULE for each run from N-tier replenishment interfaces. The former always take precedence over the latter); N (data loaded into IO_REPL_ATTR_PROD_LOC_EXP and IO_FINAL_RULE solely from RI interfaces for replenishment attributes and review schedule); X (data loaded into IO_REPL_ATTR_PROD_LOC_EXP and IO_FINAL_RULE solely through N-tier interfaces). In all cases, any rule that is set up in Rules and Strategies will override the interfaced data.

IPO Integration with RMS

When a sku-store is set up in RMS for replenishment, if the optimization flag in RMS is set to Y, the optimal replenishment policies for that sku-store is sent back to RMS via Web Service integration.

Additional configuration parameters must be set to integrate IPO with RMS. The configuration parameters that must be set for integration of replenishment policies and rebalancing transfers are listed in [Table 15-2](#) and [Table 15-3](#), respectively.

In addition, the RMS username and password must be added under the credential store for Merchandise RMS Access. You can modify these values using the UI screen for Data Management -> Manage Credential Store, as shown in [Figure 15-1](#) and [Figure 15-2](#). Data Management screen is available under the task menu in the main dashboard.

The replenishment policies and rebalancing transfers that are pushed to RMS can be viewed in io_repl_param_export_vw and io_rebalance_export_vw, respectively.

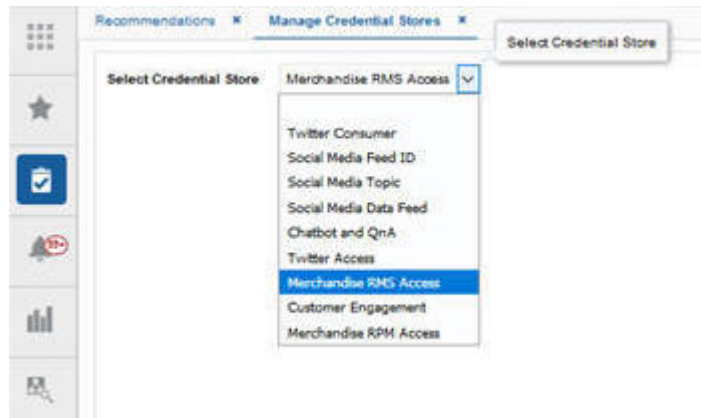
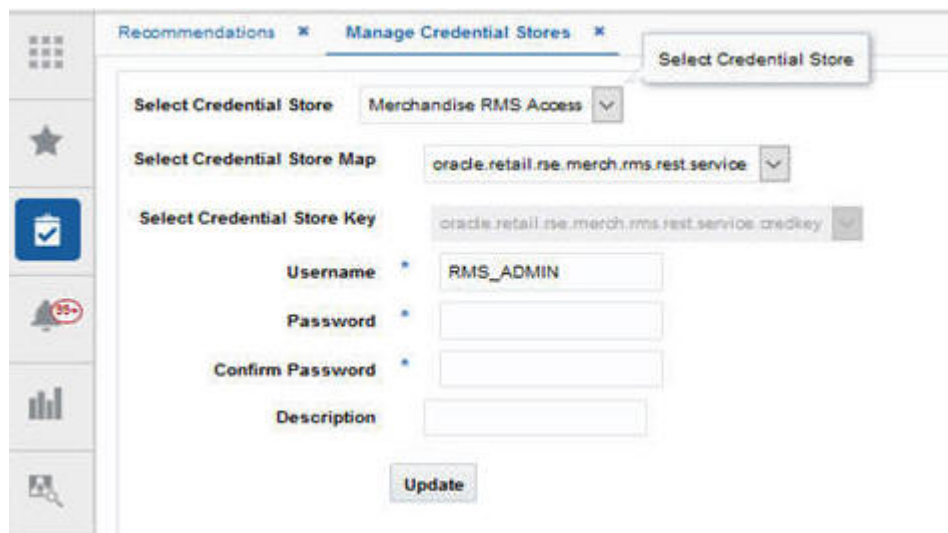
Table 15-2 Configuration Parameters for Integration of Replenishment Policies with RMS

Parameter Name	Parameter Value	Description
IO_MERCH_RMS_CONTEXT_ROOT	RmsReSTServices/services/private This value must be set based on the client environment.	Context root of the RMS Restservice url.
IO_MERCH_RMS_FLG	Y	Flag to identify whether to publish replenishment parameter to RMS.
IO_MERCH_RMS_HOSTNAME	msp00aay.us.oracle.com This value must be set based on the client environment.	URL host name to publish replenishment parameter to RMS. This must be same as CN name in the certificate.
IO_MERCH_RMS_PORT	8002 This value must be set based on the client environment.	Port number to publish replenishment parameter to RMS.
IO_MERCH_RMS_REPL_PATH	inventory/replenishment/createReplSched	Path of the Restservice e.g., inventory/replenishment/createReplSched.
IO_MERCH_RMS_VERSION	16.0 This value must be set based on the client environment.	RMS Version to use for publishing replenishment parameter to RMS (format 16.0).
IO_MERCH_RMS_AUTH_SERVICE_TYPE	OAUTH	Used in determining whether to use OAuth or basic authentication for IPO-IO RMS integration.

Table 15-3 Configuration Parameters for Integration of Rebalancing Transfers with RMS

Parameter Name	Parameter Value	Description
IO_REBAL_TRANSF_MERCH_RMS_FLG	Y	Flag to identify whether to publish rebalancing transfers to RMS.
IO_REBAL_TRANSF_MERCH_RMS_HOSTNAME	msp00aay.us.oracle.com This value must be set based on the client environment.	URL host name to publish rebalancing transfers to RMS. This must be same as CN name in the certificate.
IO_REBAL_TRANSF_RMS_PORT	443 This value must be set based on the client environment.	Port number to publish rebalancing transfers to RMS
IO_REBAL_TRANSF_RMS_SERVICE_VERSION	v1 This value must be set based on the client environment.	Transfer Management Service Version to use for publishing rebalancing transfers to RMS (format v1).
IO_TRAN_NUM_LOWER_LIMIT	9000000000	Lower range dedicated to the ID for store-to-store transfers that are generated in IPO-IO and sent to RMS.

[Figure 15-1](#) and [Figure 15-2](#) show adding a username and password for the credential store.

Figure 15-1 Selecting the Credential Store**Figure 15-2 Adding the Username and Password for the Credential Store**

Forecast Engine Setup for IPO

The forecast engine that generates demand forecast and feeds into the IPO consists of two modules: parameter estimation and demand forecasting. Parameter estimation uses the historical data to determine the parameters, including seasonality, price elasticity, promotion elasticity, and promotion lift (or holiday lift) values. Demand forecasting leverages the parameters estimated from historical data and in-season sales data (the base demand is re-estimated on a regular basis as part of batch process) to determine the demand and generate the forecast.

To set up the parameter estimation and forecast runs and configure various settings, you can use the Manage Forecast Configurations functionality under Strategy & Policy Management. Based on the configuration settings, parameter estimation can be executed through the UI to initially estimate the parameters and then updated at specified intervals to reflect the latest sales trends. Similarly, a weekly batch job is used to generate demand forecasts every week after the weekly data has been updated.

For details regarding setting up run types and runs and configuring parameters, refer to the Control and Tactical Center chapter in the implementation guide.

Here are the points to take into consideration when setting up forecast runs types for IPO.

- Forecast level: data must be aggregated at the lowest level (Sku/store/week level) and the option for spreading the forecast to day level must be enabled when setting up the run type. If customer segment data is available and is desired to be used as additional dimension in the forecast, enable the option for customer segments.
- Forecast method: if the items are fashion/short life cycle, select Causal-Short Life Cycle. If items are basic items which live for at least 16 weeks and often longer, select Causal-Long Life Cycle.
- Forecast measure: select Total Gross Sales Units.
- Spread Forecast to Day level: must be enabled.
- Spread Profile Level: Spread profile can be provided through `rse_fcst_spread_profile_stg.txt`. If the spread profile is not available, leave the profile level selection empty. In such case, a default uniform profile will be applied for spreading forecast from week to day level.
- Run the data aggregation in the Setup train stop.
- Once the aggregation is complete, use Test train stop to create an ad-hoc run. Select Run estimation and base demand and enable Run forecast. Once the run is successfully complete, if desired, you can review the results using Innovation Workbench.
- Use the Manage train stop to activate the run type and override the configurations for the run type if desired. In addition, turn on auto-approve batch runs.
- Use Map train stop to map the run type to IPO.
- Make sure proper POM jobs (listed in Batch and Ad-Hoc Jobs for IPO-DF Forecasting in the Control & Tactical Center chapter) are enabled so the forecast is generated via batch jobs.

Batch and Ad Hoc Jobs for IPO

The following jobs with mentioned options must be executed to load required data for IPO and forecast engine.

Table 15-4 Batch and Ad Hoc Jobs for IPO

JobName	Description	RmsBatch	Parameters
RSE_MASTER_ADHOC _AIF_JOB is	Run RSE master script The batch job	rse_master.ksh	You can run this with parameter '-A' which will run all data loads. You can also provide parameters to load certain data only. These parameters can be found in the AIF Operations Guide. The parameters that are relevant for IPO-IO are listed below this table.

Table 15-4 (Cont.) Batch and Ad Hoc Jobs for IPO

JobName	Description	RmsBatch	Parameters
IO_MASTER_ADHOC_AIF_JOB	Run IPO-IO master script	io_master.ksh	You can run this with parameter '-A' which will run all data loads. Alternatively, you can specify specific parameters to load certain data. List of parameters are listed below this table
IO_WAREHOUSE_LOAD_START_JOBIO_WAREHOUSE_LOAD_END_JOB	Load job for warehouses	io_warehouse_load.ksh	
IO_SEASON_LOAD_START_JOBIO_SEASON_LOAD_COPY_JOBIO_SEASON_LOAD_END_JOB	Load job for seasons	io_season_load.ksh	
RSE_RULES_TO_IO_LOAD_START_JOB RSE_RULES_TO_IO_LOAD_END_JOB	Job to load RSE rules to IPO-IO	rse_rul_eng_io_load.ksh	
IO_CREATE_BATCH_RUN_ADHOC_JOB	IPO-IO batch run creation ad hoc job	io_create_batch_run.ksh	
IO_RUN_ANALYTICS_ADHOC_JOB	IPO-IO analytics run ad hoc job	io_run_analytics.ksh	
IO_PUBLISH_MERCH_RMS_START_JOBIO_PUBLISH_MERCH_RMS_JOBIO_PUBLISH_REBAL_RMS_JOBIO_PUBLISH_REBAL_RMS_END_JOB	Publish job for pushing replenishment parameters and Rebalancing transfers to RMS	io_publish_merch_RMS.ksh io_publish_rebal_transfer.ksh	

Parameters for running RSE_MASTER_ADHOC_AIF_JOB

Table 15-5 Parameters for running RSE_MASTER_ADHOC_AIF_JOB

Data (-<parameter>)	Notes
RSE Product Hierarchy ETL (including CM Groups Hierarchy) (-p)	
RSE Location Hierarchy ETL (-l)	
Trade Area alternate hierarchy (-t)	
RSE Calendar Hierarchy ETL (-d)	

Table 15-5 (Cont.) Parameters for running RSE_MASTER_ADHOC_AIF_JOB

Data (-<parameter>)	Notes
RSE Promotion Hierarchy ETL (-r)	Required if client has promotion data that they want to use in forecasting
RSE Customer Segment Hierarchy ETL (-g)	
RSE Product Attributes (-P)	
Location attributes (-L)	
Like Location / Product data load (-K)	Required if client wants to use like location and/or like product in forecasting
Holiday load (-h)	
Inventory data load (-i)	
Sales transaction data (-x)	
Weekly Aggregate Sales data (Load or Calc) (-w)	
Aggregate Sales data processing (-a)	
Price and Cost data load (-C)	
UDA Load (-u)	
Weekly Return transactions (-T)	
Weekly Return Aggregation (-e)	
Weekly Sales Return Price Consolidation (-S)	
Promotion data load (-n)	
Daily data load (-D)	
Supplier, supplier item, daily supplier cost and supplier inventory management ETLs (-U)	w_party_org_d, w_party_attr_d, w_rtl_it_supplier_dsupplier inventory management is optional data for order scaling rules, which are sourced from w_rtl_repl_sup_mgmt_d
Seasons, phases and related items ETL (-N)	
Load rules engine data to IPO-IO (-j)	Optional. It's needed for loading IPO-IO rules that are defined in Rules & Strategies into IPO-IO.
Buyer, allocation, purchase order and transfer ETLs (-H)	Optional If provided it can be used for value assessment to compare optimized history with actual history
load forecast spread profiles (-M)	Optional. Source: rse_fcst_spread_profile_stg.txtIt's needed for spreading week level forecast to day level. If not provided, a uniform profile will be applied.
Load life cycle classification data (-V)	Optional Source: rse_fcst_life_cycle_clsif_stg.txtThis allows user to classify product/locations as being short or long life cycle. By default, all product/location combinations are considered to have the life cycle classification which is indicated by DFLT_LIFE_CYCLE in RSE_CONFIG. Exceptions to DFLT_LIFE_CYCLE can be provided through rse_fcst_life_cycle_clsif_stg.txt and loaded by running this job. This information is forecast engine to apply the proper forecast method according to the life cycle of the item.

Parameters for running IO_MASTER_ADHOC_AIF_JOB

Table 15-6 Parameters for running IO_MASTER_ADHOC_AIF_JOB

Data (-<parameter>)	Notes
Replenishment Attributes at Product/Location or Group level (-a)	Source: w_inventory_product_attr_d
Supply chain for Distribution Networks (-C)	Source: REPL_DISTRO.csv. See the RAP Foundation Data section for more details
Review cycles (-S)	Source: REPL_REV_INT.csv. See the RAP Foundation Data section for more details.
Lead times (-L)	Source: REPL_LT_INT.csv. See the RAP Foundation Data section for more details.
Supply chain for Procurement Networks (-P)	Source: REPL_PROC.csv. See the RAP Foundation Data section for more details.
Supplier Lead times (-T)	Source: REPL_LT_SUP.csv. See the RAP Foundation Data section for more details.
Internal Order Multiples (-M)	Source: REPL_MULT_INT.csv. See the RAP Foundation Data section for more details.
Supplier Order Multiples (-m)	Source: REPL_MULT_SUP.csv. See the RAP Foundation Data section for more details.
Rounding Levels (-l)	Source: REPL_ROUND.csv. See the RAP Foundation Data section for more details.
Load rules from IO table to rules engine (-l)	Source: IO_STRATEGY_RULE
Load N-tier interface data to rules engine (-N)	Source: the 8 REPL_* interfaces. See the RAP Foundation Data section for more details.
Location Priority Groups (-G)	Source: io_loc_priority_grp_stg.txt

Inventory Planning Optimization-Inventory Planning

This chapter provides details about the implementation of Inventory Planning Optimization-IP.

Security

User roles are used to set up application user accounts through Oracle Identity Management (OIM). See the *Retail Analytics and Planning Solutions Administration Guide* for details. One of the following roles is required to access the IPO-IP application:

- IP_SYSTEM_ADMINISTRATOR in order to access the IPO-IP Application Properties (Settings > Configure > Application Properties)
- IP_INVENTORY_PLAN_VIEWER in order to view the IPO-IP Plan View
- IP_INVENTORY_PLAN_EDITOR in order to edit policies on the Advanced Options panel

In addition to above roles, verify the following:

- Access to Data Visualizer: This is necessary in order to query or visualize the data and verify that the data loaded matches the desired expectations.
- Access to POM to execute ad-hoc and batch jobs: The POM UI URL is something like <host>/POMJetUI. If the user cannot access the POM UI, contact the administrator to obtain the relevant access/user roles.
- IP_DV_VIEWER role is required in order to access the Oracle Analytics link on Retail Home (links to Data Visualizer).

Requirements

In order to load data into the IPO-IP Plan View, IPO-IO must be fully set up. IO data must be loaded, and the IO_RUN_ANALYTICS_ADHOC_JOB (IO Batch Job) must have completed successfully. Then IP batch processing is necessary to build tables and data that will be used to render the Plan View.

Batch Processing

IPO-IP batch jobs are managed by Process Orchestration and Monitoring (POM). IPO-IP batch is executed after an IO batch successfully completes execution.

Batch and Ad Hoc Jobs for IPO-IP

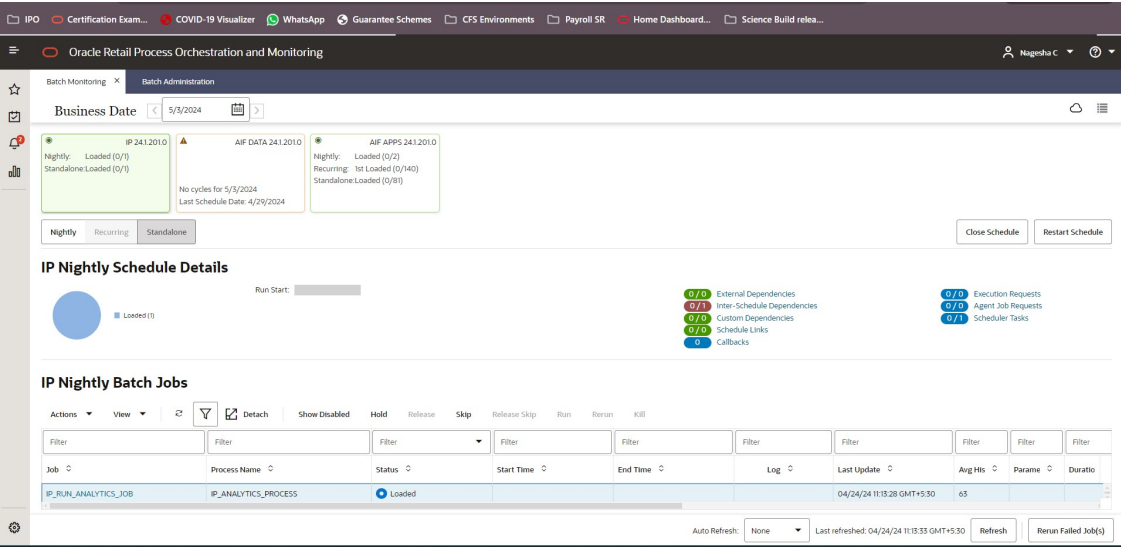
The IP_RUN_ANALYTICS_JOB batch job must be executed to load data into fact tables used to render the Inventory Plan View, as shown in the following table.

Job Name	Description
IP_RUN_ANALYTICS_JOB	Runs the IP_RUN_ANALYTICS_PROCESS in POM. Performs the following actions: • Rebuild Plan View Fact Tables - Refreshes/ rebuilds fact tables used to render the Inventory Plan View. • Generate Alerts - Generates Inventory Planning alerts based on IO Strategy rules.

Batch Monitoring for IPO-IP

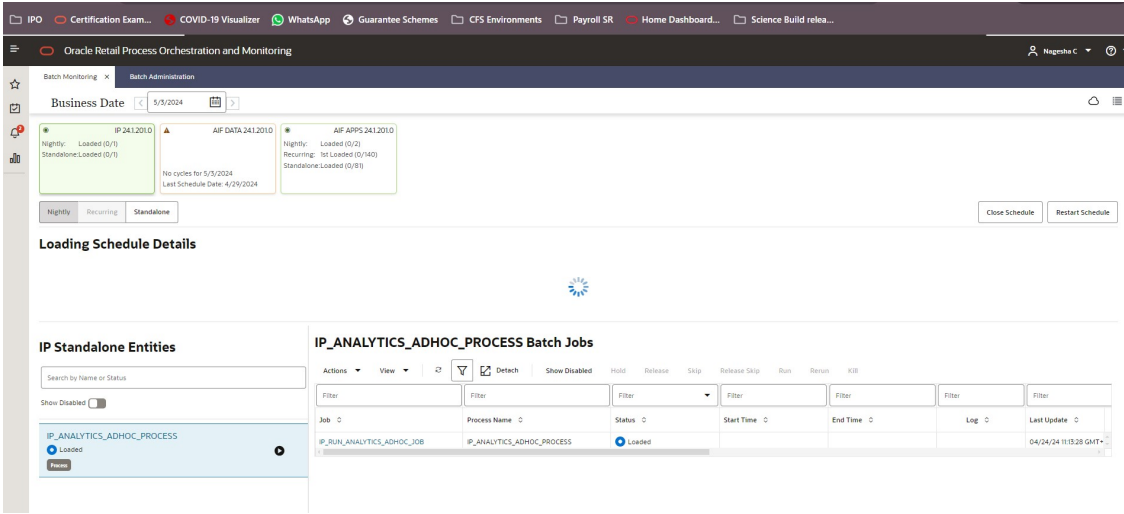
In POM's Batch Administration, IP batch jobs are accessible from the IP Tile. IP_RUN_ANALYTICS_JOB has an Inter-Schedule dependency on IO_RUN_ANALYTICS_JOB. A nightly schedule can be created in Batch Monitoring to ensure that IP_RUN_ANALYTICS_JOB is executed after IO_RUN_ANALYTICS_JOB successfully completes execution.

Figure 16-1 Batch Monitoring in POM — IP Nightly Batch Job



An IP Standalone adhoc job is provided, as shown in the following figure.

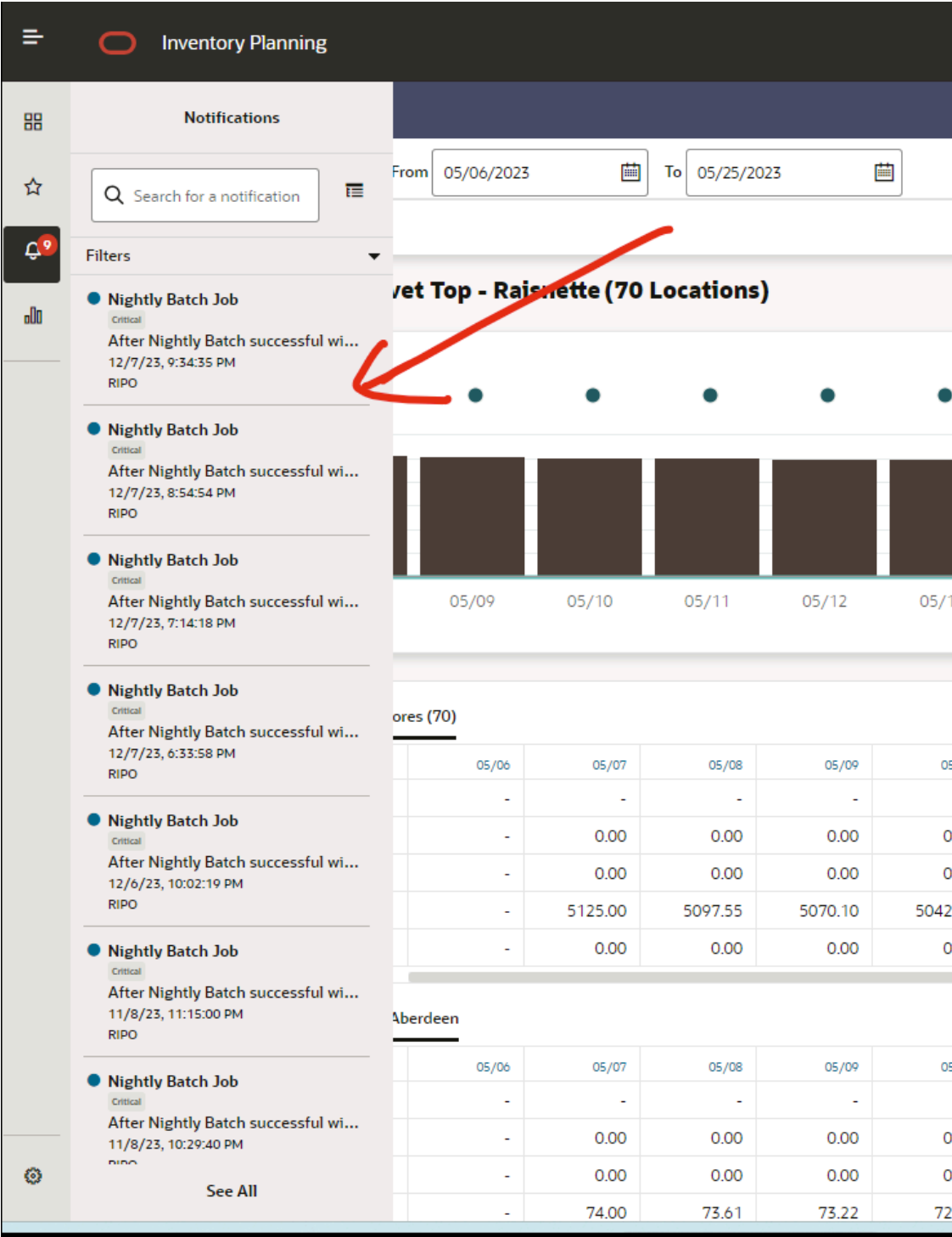
Figure 16-2 IP Batch Monitoring in POM — IP Standalone Adhoc Job



Batch Monitoring for IPO-IP - Status Notifications

IP System Administrators are notified when a batch run completes. A notification indicates whether the batch run completed successfully or failed, as shown in the following figure.

Figure 16-3 IP Batch Job Notifications



17

Configuration

This chapter describes the major application configuration points, including:

- [User Interface Authentication and Authorization](#)
- [User Management Configuration: Configuring Users and Roles](#)
- [Configuration](#)
- [Internationalization](#)
- [Affinity Analysis Configurations](#)

Note

Since AA is distinct from the other applications, much of what is described here is not applicable for AA. For clarity, AA implementation, configuration, operations and data model are described separately in [Affinity Analysis](#).

User Interface Authentication and Authorization

Note

For more information, see *Oracle Retail AI Foundation Cloud Services Administration Guide*.

For authorization, the applications have been built with role-based access. Access to application user interface components is done by assigning application roles. Application roles are defined as part of the application and deployed as part of the installation process. Application roles are mapped to enterprise roles during initial environment provisioning. Enterprise roles exist as LDAP groups in OID. Refer to the *Oracle Retail AI Foundation Cloud Service User Guide* for the definition of standard user roles.

User Management Configuration: Configuring Users and Roles

This section provides detailed instructions on setting up enterprise-level user management using Oracle WebLogic 12c with Enterprise Manager. The user management configuration is handled using the WLS Console and the WLS Enterprise Manager (EM).

User Roles

The roles listed in [Table 17-1](#) are required to update the configurations in the data management section of the application. For more information, see *Oracle Retail AI Foundation Cloud Services Administration Guide*.

Table 17-1 Data Management Configuration Roles

Roles	Access to Application Configuration
ANALYTIC_EXPERT_JOB	CDT
ANALYTIC_EXPERT_JOB	DT
CLUSTERING_ADMINISTRATOR_JOB	AC
CUSTOMER_SEGMENT_ADMINISTRATOR_JOB	CS
RETURN_LOGISTICS_JOB	RL
SOCIAL_ANALYTICS_JOB	SA
PRICING_ADMINISTRATOR_JOB	PRO
MARKET_BASKET_ANALYSIS_JOB	MBI

Configuration

This section provides details about application configurations that can be modified as part of a deployment. The list of configurations is limited to those settings that most likely need to be reviewed and adjusted before the applications are used. Some configuration points cannot be adjusted after initial setup, so a careful review of the configurations should be done. For a complete list of configurable values, see the complete list of configuration values in RSE_CONFIG.

Any installation configuration must be set before the application is initialized with data and used.

Note

Before beginning any advanced customization, you must consult with development.

RSE_CONFIG Table

The RSE_CONFIG table is a generic table that contains various application configurations. The DESCR column describes the configuration and provides any relevant information regarding what each specific configuration controls. The APPL_CODE column is a useful filter to organize the configurations by the application module.

The following application abbreviations are used:

- CDT - Customer Decision Trees
- CIS - Advanced Clustering
- DT - Demand Transference
- IO -Inventory Planning Optimization
- MBA - Affinity Analysis
- PMO - Forecasting
- PRO - Offer Optimization
- RL - Return Logistics

- RSE - Retail Science Platform (generic configurations)
- SA - Social Analytics
- SPO - Size Profile Optimization
- SO - Space Optimization
- The PARAM_NAME is the name of the configuration.
- The PARAM_VALUE is the value currently associated with the configuration.
- The CONFIGURABLE_FLG column indicates whether the configuration is safe to change.
- The UPDATEABLE_FLG column is used to control whether application code is able to update the configuration.

Note

Use caution when changing a value that the application code can update after the application has started to be used.

The RSE_CONFIG_CODE table is an extension table to the RSE_CONFIG table. It allows for overrides of certain configurations when an additional qualifier is present. All descriptions for the columns in RSE_CONFIG apply here as well. The additional column CONFIG_CODE allows overriding a value for a specific qualifier. Application code does not always support the use of these configurations, but Affinity Analysis, for example, uses this to set up configurations for different hierarchy levels.

Generic Configuration

The following is a list of configurations that can be adjusted. This list does not include internal configuration values. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_CODE='RSE'.

Table 17-2 Generic Configurations

Parameter Name	Example Value	Description	Notes
CAL_PERIOD_LEVEL	4	This is the calendar hierarchy level that is used to drive the processing (installation configuration).	Usually this value should be set to the Fiscal Week level of the Fiscal Calendar. The number 4 refers to the fourth level in RSE_HIER_LEVEL table. This cannot be adjusted after initial setup.

Table 17-2 (Cont.) Generic Configurations

Parameter Name	Example Value	Description	Notes
CMGRP_HIER_TYPE	1	The hierarchy ID to use for the CM Group (installation configuration).	If an alternate hierarchy is to be used, then this configuration must specify that hierarchy type. Otherwise it must reflect the primary product hierarchy type. The value represented here relates to the ID in RSE_HIER_TYPE. This cannot be adjusted after initial setup.
CMGRP_LEVEL_ID	5	The hierarchy level ID that contains the level of the product hierarchy where the CM Group level exists (installation configuration).	Once the CMGRP_HIER_TYPE is configured, this level must be set to indicate the level of the hierarchy (RSE_HIER_LEVEL) that defines the categories.
PRIMARY_LANGUAGE_CODE	EN	The name of the language code to use for all RSE data sourced from AIF data warehouse tables (installation configuration).	Data values stored in the database are not multi-language capable, and are not affected by the UI language settings, like the UI labels are. This setting must select the language code for which data should be shown in the UI.
RA_FISCAL_CAL_ID	1240	ID of the calendar to use from AIF data warehouse since it supports multiple calendars (installation configuration).	This value must match the ID of the desired fiscal calendar loaded into W_MCAL_PERIOD_DS.MCAL_CAL_WID.
CHAIN_LEVEL_DESC	CHAIN	The description to use for any top level hierarchy element when one must be manually created.	This description must be adjusted if a different description for the top level of the hierarchy is desired.
DEFAULT_LOCALE	en_US	The default locale to use for rendering elements that cannot support multiple locales.	Adjust this value so that it contains the correct LOCALE setting for most users of the application.
DISPLAY_DATE_FORMAT	Mon dd, yyyy	The default date mask to be used by UI.	Adjust this so that the format of dates can be displayed as desired.

Table 17-2 (Cont.) Generic Configurations

Parameter Name	Example Value	Description	Notes
FAKE_CUST_DAY_TXN_TH RESHOLD	10	The maximum daily transaction threshold to identify a fake customer.	This setting must be adjusted to the maximum number of daily transactions that a normal customer usually has. This allows all other customers with higher transactions counts to be excluded from processing.
UI_TZ	America/ New_York	Timezone for display. Must match SELECT tzname FROM V\$TIMEZONE_NAMES.	This setting must be adjusted so that it contains a proper time zone setting indicating where most users run the UI.

Advanced Clustering Configurations

The following is a list of configurations that can be adjusted for the Advanced Clustering application. This list does not include internal configuration values. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_CODE='CIS'.

Store Exclusion can be used to filter and identify which stores are to be excluded for Clustering. To enable this functionality, EXCLUDE_LOC_ATTR is set by specifying a SHORT_DB_NAME from the RSE_BUSINESS_OBJECT_ATTR_MD BOAM) table in RSE_CONFIG. This can be configured via the Control and Tactical Center. Locations in RSE_LOC_ATTR table with the BUSINESS_OBJECT_ATTR_MD_ID value similar to the ID of from BOAM table, which is configured in the config table, will be excluded.

For example, EXCLUDE_LOC_ATTR of RSE_CONFIG is set to CFA_ATTRIB_2345 as its value. A check is performed to fetch the ID from the BOAM table that contains this SHORT_DB_NAME. This ID is then used to obtain all locations from the RSE_LOC_ATTR table. These locations are then flagged to be excluded from clustering.

Table 17-3 AC Configurations

Parameter Name	Example Value	Description	Notes
PERF_CIS_APPROACH	CDT	The approach to use for performance-based clustering. Available options are CDT and DT.	If the CDT application is in use, then this configuration must be set to CDT, which will result in the use of attribute groups for product attribute clustering. Otherwise, it must be set to DT, which will result in the use of the top raw attribute values. This cannot be adjusted after initial setup.

Table 17-3 (Cont.) AC Configurations

Parameter Name	Example Value	Description	Notes
ATTR_NAME_SEPARATOR	-	The separator character(s) to use to separate the different components of the attribute names in CIS_TCRITERIA_ATTR.	Adjust this value if a different separator is desired in the UI for attribute names built from multiple values. This cannot be adjusted after initial setup.
CIS_CONTR_SLS_SRC_COLUMN	SLS_AMT	Source column for contribution BI chart sales values. SLS_AMT, SLS_QTY, PROFIT_AMT are the allowable values.	Adjust this value as needed so that the BI shows contributions based on the desired sales column.
CIS_CONTR_X_SRC_COLUMN	SLS_AMT	Source column for the x axis of the contribution BI chart. SLS_AMT, SLS_QTY, PROFIT_AMT are the allowable values.	Adjust this value as needed so that the BI shows contributions based on the desired sales column.
CIS_CONTR_Y_SRC_COLUMN	SLS_QTY	Source column for the y axis of the contribution BI chart. SLS_AMT, SLS_QTY, PROFIT_AMT are the allowable values.	Adjust this value as needed so that the BI shows contributions based on the desired sales column.
CIS_DFT_PIVOT_LVL	6	Default pivot level to show in explore data.	This configuration indicates the lowest level of the organization hierarchy (see rse_hier_level) that should be shown in the Explore Data pivot table. A value of 6 allows store locations to be visible, but this can be adjusted to a higher level if this level of detail is not desired.
CIS_IDX_AVG_SRC_COLUMN	SLS_AMT	Name of the column to use for index-to-average BI calculations. SLS_AMT, SLS_QTY, PROFIT_AMT are the allowable values.	Adjust this value as needed so that the BI shows index to averages based on the desired sales column.
CIS_NUMERIC_DFT	0	Default attribute value for numeric.	This can be adjusted so that attributes without a value are displayed with the desired value. This cannot be adjusted after initial setup.
CIS_STRING_DFT	UNKNOWN	Default attribute value for string.	This can be adjusted so that attributes without a value are displayed with the desired value. This cannot be adjusted after initial setup.

Table 17-3 (Cont.) AC Configurations

Parameter Name	Example Value	Description	Notes
CIS_VARIABILITY_IDX_SR C_COLUMN	SLS_AMT	Name of the column to use for variability index BI calculations. SLS_AMT, SLS_QTY, PROFIT_AMT are the allowable values.	Adjust this value as needed so that the BI shows variability indexes based on the desired sales column.
DEFAULT_NUM_ATTR_VAL UE	15	Constant for number of discrete values allowed for store attribute clustering.	Any attribute that has more than this number of distinct categorical values is not available for use as an attribute.
DEFAULT_STR_CATEGORI CAL_ATTR	UNKNOWN	Default string description for row added in cis_tcriteria_attr_type_val ue table for unmatched grouping.	This can be adjusted so that attributes without a value are displayed with the desired value. This cannot be adjusted after initial setup.
EXCLUDE_LOC_ATTR	CFA_ATTR5_234 5	Contains the name of a SHORT_DB_NAME from RSE_BUSINESS_OBJECT_ATTR_MD table that is used to exclude specific stores that are in RSE_LOC_ATTR to be excluded from AC processing.	Setting this configuration to NULL disables the Store Exclusion interface.
INSIGHT_BI_SALES_AMT_ MARGIN	N	Flag to show sales BI based on sales amount and margin.	Either INSIGHT_BI_SALES_AMT_MARGIN or INSIGHT_BI_SALES_AMT_UNIT can be assigned a Y so that the BI displays the BI using the desired columns.
INSIGHT_BI_SALES_AMT_ UNIT	Y	Flag to show sales BI based on sales amount and units	-
MAX_HIST_WEEK_CNT	104	The maximum number of weeks that must be selectable by the UI when processing historic data.	-
MAX_ITEMS_IN_GRAPH_C LUSTER_DETAIL	-1	Maximum number of clusters to be displayed in cluster details graph.	A value of -1 results in an unlimited number of values. Adjust if necessary.
MAX_ITEMS_IN_GRAPH_C LUSTER_LIST	-1	Maximum number of clusters to be displayed in cluster list graph.	A value of -1 results in an unlimited number of values. Adjust if necessary.
MNG_RUN_NO_WKS	26	Display run for past n weeks.	-

Table 17-3 (Cont.) AC Configurations

Parameter Name	Example Value	Description	Notes
PERF_ATTR_TOPN_COUNT	3	The number of attribute values to be used per product category for performance-based clustering.	If PERF_CIS_APPROACH configuration is DT, then this configuration will limit the number of attribute values to this number of values with the greatest sales.
PERF_NUM_WEEKS_FOR_SLS_SHARE	16	The number of weeks to be used while calculating the sales share for the product attributes.	-
PERF_NUM_WEEKS_FOR_TOPN_CALC	16	The number of weeks to be used while identifying the top <i>n</i> attributes	-
SELECT_ALL_MERCH_NODE	N	Flag to identify if all (or only first) merchandise node(s) to be selected by default.	-
SUMM_CAL_ALL_LVL	N	Flag to identify whether performance summarization allowed at all available calendar levels.	-
SUMM_MERCH_ALL_LVL	N	Flag to identify whether performance summarization allowed at all available merchandise levels.	-

Customer Decision Tree Configurations

The following is a list of configurations that can be adjusted the CDT application. This list does not include internal configuration values. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_CODE='CDT'. See [Customer Decision Trees](#) for more details.

Table 17-4 CDT Configurations

Parameter Name	Example Value	Description	Notes
CDT_CAL_LEVEL_ID	4	The hierarchy level ID that contains the level of the calendar hierarchy that CDT operates on (should equate to Week - Installation configuration).	Normally, this configuration must be the same value as the common CAL_PERIOD_LEVEL configuration.

Table 17-4 (Cont.) CDT Configurations

Parameter Name	Example Value	Description	Notes
CDT_CMGRP_LEVEL_ID	5	The hierarchy level ID that contains the level of the product hierarchy that CDTs are created for (installation configuration).	Normally, this must be the same setting as the common CMGRP_LEVEL_ID configuration.
CDT_LOC_HIER_TYPE	2	The hierarchy ID to use for location (installation configuration).	This setting must be set to use either the trade area hierarchy type or the organization hierarchy type.
CDT_LOC_LEVEL_ID	4	The hierarchy level ID that contains the level of the location hierarchy that CDTs are created for (installation configuration).	If CDT_LOC_HIER_TYPE is set to trade area, this must be set to a value of 2. Otherwise, it must be set to the level of the organization hierarchy for which CDT output is desired.
CDT_PROD_HIER_TYPE	1	The hierarchy ID to use for the CM Group (installation configuration).	Normally, this setting must be the same value as the common CMGRP_HIER_TYPE configuration. It can either be the primary product hierarchy or an alternate product hierarchy.
CDT_CALC_RAW_ATTR_SIM	Y	Determines whether or not to execute Raw Attribute Value Similarities routines.	If the AIF data warehouse has been loaded with raw attributes, then this setting must be set to Y to enable calculation of raw attr similarities.
CDT_UI_DEF_CALC_PARENT_SEGMENT_FLG	Y	UI default for the calculate only parent consumer segments flag.	-
CDT_UI_DEF_CALC_PARENT_TRADE_AREA_FLG	N	UI default for calculate only parent trade areas flag.	-
CDT_UI_DEF_CDT_SCORE_HIST_CNT	20	UI default for the number of histogram buckets for the CDT scores histogram.	-
CDT_UI_DEF_DATA_FILTER_HIST_CNT	20	UI default for the number of histogram buckets for the data filtering histograms.	-
CDT_UI_DEF_EXCLUDE_CUSTOMER_CNT	1000	UI default for minimum require customer counts for pruning process.	-
CDT_UI_DEF_EXCLUDE_MIN_CDT_SCORE	0.25	UI default for minimum CDT score required for the pruning process.	-

Table 17-4 (Cont.) CDT Configurations

Parameter Name	Example Value	Description	Notes
CDT_UI_DEF_EXCLUDE_S KU_CNT	10	UI default for minimum number of SKUs for the pruning process.	-
CDT_UI_DEF_EXCLUDE_T REE_LEVEL_CNT	2	UI default for minimum number of levels of the tree for the pruning process.	-
CDT_UI_DEF_LOWEST_EX PANSION_LEVEL	15	UI default for lowest number of levels allowed for a tree.	-
CDT_UI_DEF_MAX_CUST_ AVG_DY_TXN	100	UI default for maximum number of times more than average a customer's daily transaction count can be.	-
CDT_UI_DEF_MAX_MISS_ ATTR_CNT	3	UI default for maximum number of missing attributes a SKU can have	-
CDT_UI_DEF_MIN_ATTR_S KU_CNT	5	UI default for minimum number of SKUs assigned to an attribute, to be used by the process.	-
CDT_UI_DEF_MIN_ATTR_V ALUE_SKU_CNT	5	UI default for minimum number of SKUs assigned to an attribute value, to be used by the process.	-
CDT_UI_DEF_MIN_CUST_ TXN_CNT	0.01	UI default for minimum number of transactions required for a customer, as a percent of the average number.	-
CDT_UI_DEF_MIN_NODE_I TEM_CNT_PCT	0.05	UI default for the minimum percent of SKUs required for a node of the tree before it is considered a terminal node.	-
CDT_UI_DEF_MIN_SKU_T XN_CNT	0.01	UI default for minimum number of transactions required for a SKU, as a percent of the average number.	-
CDT_UI_DEF_PRUNING_HI ST_CNT	20	UI default for the number of histogram buckets for the pruning histograms.	-
CDT_XML_PRECISION	4	Default precision of weight field in CDT XML.	- Adjust this to control the amount of precision in the generated CDT XML files.
HISTOGRAM_DEFAULT_BI N_APPROACH	C	The default histogram bin approach (C = Custom, W = Width)	-

Table 17-4 (Cont.) CDT Configurations

Parameter Name	Example Value	Description	Notes
HISTOGRAM_DEFAULT_NUM_BINS	7	The default number of bins to display for CDT histograms	-
MAX_NUM_WEEKS_FOR_SIMILARITY	104	The maximum number of weeks of sales transaction data to be used by the similarity process. This prevents the process from using too much data.	Adjust this to reduce the use of too much input data for the process. Enabling a high number of weeks will result in slower performance; however, it may be necessary if suitable data is not available in a smaller number of weeks.

Demand Transference Configurations

The following is a list of configurations that can be adjusted for the DT application. This list does not include internal configuration values. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_CODE='DT'.

Table 17-5 DT Configurations

Parameter Name	Example Value	Description	Notes
AE_CALC_INT_LENGTH	8	The number of weeks to group together for in an interval for the AE calculation.	This setting must be adjusted according to the data quality and quantity available for a customer. Higher values improve performance; however, it limits the amount of data available to be processed.
ATTRIBUTE_LIST_SEPARATOR	-	A separator to be used to display a list of attributes in Similarity Calculation screen.	-
CDT_SIMILARITY_AVAILABLE	Y	Whether CDT similarity has been made available to DT.	If the CDT application is being used, this setting enables DT to use the similarities that CDT may have calculated.
DT_CAL_LEVEL_ID	4	The hierarchy level ID that contains the level of the calendar hierarchy that DT operates on (should equate to Week).	Normally, this configuration must be the same value as the common CAL_PERIOD_LEVEL configuration.
DT_CMGRP_LEVEL_ID	5	The hierarchy level ID that contains the level of the product hierarchy that DTs are created for.	Normally, this must be the same setting as the common CMGRP_LEVEL_ID configuration.

Table 17-5 (Cont.) DT Configurations

Parameter Name	Example Value	Description	Notes
DT_LOC_HIER_TYPE	2	The hierarchy ID to use for location.	This must be adjusted to trade area hierarchy or be left at the default organization hierarchy.
DT_LOC_LEVEL_ID	4	The hierarchy level ID that contains the level of the location hierarchy that DTs are created for.	If DT_LOC_HIER_TYPE is set to trade area, this must be set to a value of 2. Otherwise, it must be set to the level of the organization hierarchy for which DT output is desired.
DT_MDL_AP_EXP_WKS_B ACK_END	1	The number of weeks back from the last date that range data has been loaded for (PR_LOC_STATUS_LAST_COMPLETED_WK) to end using for model apply export.	Adjust this and MDL_AP_EXP_WKS_BA CK_START to control which weeks should be used during data export.
DT_PROD_HIER_TYPE	1	The hierarchy ID to use for the CM Group.	Normally, this setting must be the same value as the common CMGRP_HIER_TYPE configuration. It can either be the primary product hierarchy or an alternate product hierarchy.
DT_REMOVE_REDUNDAN CY	N	If set to Y, then remove redundancy while calculating attribute-based similarities.	-
DT_SIM_DISPLAY_ROWNU M	9999999	The number of distinct similarity values to show in the UI pop-up. Setting to a high number effectively eliminates this limit.	-
GENERIC_SEPARATOR		A separator to be used to display a list of items, for example. SKU prod_ext_code name.	This value is used in the UI to separate lists of items. For example, when a list of attributes is shown in the UI, they will be delimited by this value.
HISTOGRAM_DEFAULT_BI N_APPROACH	W	The default histogram bin approach (C = Custom, W = Width).	-
HISTOGRAM_DEFAULT_NU M_BINS	7	The default number of buckets in the contextual BIs.	-

Table 17-5 (Cont.) DT Configurations

Parameter Name	Example Value	Description	Notes
MAX_NUM_WEEKS_FOR_ATTR_WGT	104	The maximum number of weeks of input data to use for calculating attribute weights.	Setting this value to a higher value allows the use of more weeks of data, which will slow performance; however, enables better results for categories with infrequent sales.
MAX_NUM_WEEKS_FOR_AVG_SLS	104	The maximum number of weeks of input data to use for calculating the average sales.	Setting this value to a higher value allows the use of more weeks of data, which will slow performance; however, enables better results for categories with infrequent sales.
MAX_NUM_WEEKS_FOR_FILTERING	104	The maximum number of weeks of input data to use for data filtering. Setting this value lower than the other MAX_NUM_WEEKS_FOR* configurations overrides those other configurations.	Setting this value to a higher value allows the use of more weeks of data, which will slow performance; however, enables better results for categories with infrequent sales.
MAX_NUM_WEEKS_FOR_MDL_CALC	104	The maximum number of weeks that should be used during model build calculation.	Setting this value to a higher value allows the use of more weeks of data, which will slow performance; however, enables better results for categories with infrequent sales.
MAX_NUM_WEEKS_FOR_MDL_UPDT	104	The maximum number of weeks that should be used during model build update calculation.	Setting this value to a higher value allows the use of more weeks of data, which will slow performance; however, enables better results for categories with infrequent sales.
MAX_NUM_WEEKS_FOR_SIMILARITY	104	The maximum number of weeks of input data to use for calculating similarity.	Setting this value to a higher value allows the use of more weeks of data, which will slow performance; however, enables better results for categories with infrequent sales.

Table 17-5 (Cont.) DT Configurations

Parameter Name	Example Value	Description	Notes
MDL_AP_EXP_WKS_BACK _START	4	The number of weeks back from the last date that range data has been loaded for (PR_LOC_STATUS_LAST_COMPLETED_WK) to start using for model apply export.	Adjust this and DT_MDL_AP_EXP_WKS_BACK_END to control which weeks should be used during data export.
PRUNED_CATEGORIES_S EPARATOR	,	A separator to be used to display a list of pruned attributes in the Calculation screen.	-
UI_DEF_CALC_PARENT_C S_ONLY_FLG	N	The UI default for calculate only parent customer segments flag.	-
UI_DEF_CALC_PARENT_T A_ONLY_FLG	N	The UI default for calculate only parent trade areas flag.	-
UI_DEF_MAX_MISS_ATTR_ CNT	3	The maximum number of missing attributes for a SKU, before requiring it to be filtered from use.	-
UI_DEF_MIN_SKU_CNT	10	The UI default for minimum number of SKUs required for a segment/store.	-
UI_DEF_MIN_SKU_TXN_LE N_PCT	0.01	The UI default for minimum SKU transaction length as a percentage of the CM Group average.	-
UI_DEF_MIN_TOT_SLS_UN IT_PCT	0.01	The UI default for minimum total sales units as a percentage of the CM group average.	-
WGT_CALC_INTERVAL_LE NGTH	4	The number of weeks to group into an interval that is then used to perform weight calculations with.	This setting can be adjusted according to the data quality and quantity available for a customer. Higher values improve performance; however, it limits the amount of data available to be processed.

Returns Logistics Configurations

The following is a list of configurations that can be adjusted for the Returns Logistics process. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_CODE='RL'.

Table 17-6 Returns Logistics Configurations

Parameter Name	Example Value	Description	Notes
RL_NUM_TOP_CATEGORIES	10	The number of top selling categories for Returns Logistics to process.	-
RL_NO_OF_SIMULATION_RUNS	100	The number of simulated forecast runs to perform.	-
RL_IN_SEASON_DEMAND_WEIGHT	0.99	The percentage of the in-season demand and returns rate to use. This is used to calculate the adjusted demand and returns rate.	-
RL_HISTORICAL_DEMAND_WEIGHT	.01	The percentage of the historical demand and returns rate to use. This is used to calculate the adjusted demand and returns rate.	-
RL_GUR_TIMELIMIT	60	GUROBI parameter. Time limit.	-
RL_GUR_THREADS	0	GUROBI parameter. Number of allowed threads.	-
RL_PROCESSING_THREADS	0	The number of threads to create for data processing. A value of 0 creates one thread for each machine processor.	-

Affinity Analysis Configurations

The following is a list of configurations that can be adjusted for the AA application. This list does not include internal configuration values. All of these entries are adjusted by manipulating the corresponding rows in RSE_CONFIG where APPL_CODE='MBA'. Some of the configurations note that overrides are possible. When this is required, see the RSE_CONFIG_CODE configuration table and use the appropriate Hierarchy Level value as the PARAM_CODE to override the value for only that hierarchy level.

Table 17-7 AA Configurations

Parameter Name	Example Value	Description	Notes
ARM_CS_HIER_LEVEL	SBC	The highest level of the hierarchy to run the process for. Valid values are: CLS, SBC.	-
ARM_CS_MAX_LIFT	100	The maximum lift required for an association rule.	-

Table 17-7 (Cont.) AA Configurations

Parameter Name	Example Value	Description	Notes
ARM_CS_MAX_RULE_CO UNT	9999	The maximum number of rules desired for association rules.	Affects the count of rules per set size.
ARM_CS_MAX_SET_SIZE	2	The maximum number of hierarchy members to include in an association rule.	Maximum allowed is 4, although a setting this high can negatively affect performance with some datasets.
ARM_CS_MIN_CONFIDEN CE	.05	The minimum confidence required for an association rule.	-
ARM_CS_MIN_LIFT	.05	The minimum lift required for an association rule.	-
ARM_CS_MIN_REV_CONF IDENCE	.05	The minimum reverse confidence required for an association rule.	-
ARM_CS_MIN_SUPPORT	.001	The minimum percentage of transactions require for an association rule.	-
ARM_CS_MIN_SUPPORT_ TXN_CNT	1000	The minimum number of sales transactions required for creating association rules.	-
ARM_CS_WEEK_CNT	1	The number of weeks that should be processed while calculating the association rules.	Only set this to more than 1 if there is a need to reprocess weeks that would have been processed in the prior batch window.
ARM_PH_MAX_LIFT	100	The maximum lift required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PH_MAX_RULE_CO UNT	9999	The maximum number of rules desired for association rules.	Affects the count of rules per set size.
ARM_PH_MAX_SET_SIZE	2	The maximum number of hierarchy members to include in an association rule. Override with PARAM_CODE of the hierarchy level name.	Maximum allowed is 4, although a setting this high can negatively affect performance with some datasets.
ARM_PH_MIN_CONFIDEN CE	.05	The minimum confidence required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-

Table 17-7 (Cont.) AA Configurations

Parameter Name	Example Value	Description	Notes
ARM_PH_MIN_LIFT	.05	The minimum lift required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PH_MIN_REV_CONFIDENCE	.05	The minimum reverse confidence required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PH_MIN_SUPPORT	.001	The minimum percentage of transactions require for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PH_MIN_SUPPORT_TXN_CNT	1000	The minimum number of sales transactions required for creating association rules.	-
ARM_PH_TOP_LEVEL	CLS	The highest level of the hierarchy to run the process for. Valid values are: DEPT, CLS, SBC.	-
ARM_PH_WEEK_CNT	1	The number of weeks that should be processed while calculating the association rules.	Only set this to more than 1 if there is a need to reprocess weeks that would have been processed in the prior batch window.
ARM_PROMO_HIER_LEVEL	SBC	The highest level of the hierarchy to run the process for. Valid values are: CLS, SBC.	-
ARM_PROMO_MAX_LIFT	100	The maximum lift required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PROMO_MAX_RULE_COUNT	9999	The maximum number of rules desired for association rules.	Affects the count of rules per set size.
ARM_PROMO_MAX_SET_SIZE	2	The maximum number of hierarchy members to include in an association rule. Override with PARAM_CODE of the hierarchy level name.	Maximum allowed is 4, although a setting this high can negatively affect performance with some datasets.
ARM_PROMO_MIN_CONFIDENCE	.05	The minimum confidence required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-

Table 17-7 (Cont.) AA Configurations

Parameter Name	Example Value	Description	Notes
ARM_PROMO_MIN_LIFT	.05	The minimum lift required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PROMO_MIN_REV_CONFIDENCE	.05	The minimum reverse confidence required for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PROMO_MIN_SUPP_ORT	.001	The minimum percentage of transactions require for an association rule. Override with PARAM_CODE of the hierarchy level name.	-
ARM_PROMO_MIN_SUPP_ORT_TXN_CNT	1000	The minimum number of sales transactions required for creating association rules.	-
ARM_PROMO_WEEK_CNT	1	The number of weeks that should be processed while calculating the association rules.	Only set this to more than 1 if there is a need to reprocess weeks that would have been processed in the prior batch window.

Resource Bundles

Oracle Retail AI Foundation Cloud Services comes packaged with resource bundles, files that contain text resources. These text resources appear throughout the application as instructions, messages, labels, errors. Retailers can customize the text resources to suit their requirement. The capability to change the text for a resource is provided through the Resource Bundles Section of Retail Home. For more information, see *Oracle Retail Home Administration Guide*.

Manage Notebooks

Different types of notifications are sent to specific groups of users for the tasks and jobs accomplished. Retail Home provides administrators with the ability to set up groups of user roles that must receive each type of notification. For more information, see *Oracle Retail Home Administration Guide*.

Configurations for Loading Fact Measures into AIF

This section covers the details of setup and configurations in the case where fact data is provided at an aggregated level, for example, where sales units data is provided at sku-store-week level or any higher level such as department-region-week (as opposed to sales transaction).

Interfaces for Fact Measures

The following four interfaces allow for providing data at a maximum of four different intersections of merchandise, location, and calendar. For details about configurations and loading data via these interfaces, see the *Oracle Retail Analytics and Planning Implementation Guide*.

Table 17-8 Interfaces for Fact Measures

Interface Name	Staging Table in Data Exchange (RDX) Layer	Target Table in AIF Data Warehouse
Fact Measures	W_PDS_ACTUALS_STG1	W_RTL_FACT_PROD1_LC1_T1_FS
	W_PDS_ACTUALS_STG2	W_RTL_FACT_PROD2_LC2_T2_FS
	W_PDS_ACTUALS_STG3	W_RTL_FACT_PROD3_LC3_T3_FS
	W_PDS_ACTUALS_STG4	W_RTL_FACT_PROD4_LC4_T4_FS

Configurations in AIF for Loading FACT Measures

The fact data can be at any intersection and can contain various measures, such as sales, inventory, shipment, and so on.. Since the intersections and measures vary across customers, it is important to configure AIF for each implementation based on the intersections and measures that are planned to be loaded. The following tables are used for specifying the intersections, measures, and mappings.

- RSE_MD_CDA:** This table contains SRC_ENTITY_NAME, SRC_COLUMN_NAME, and TGT_COLUMN_NAME. The SRC columns have the name of the FACT table and name of the column from that FACT table that are to be loaded into AIF. The TGT column has the name of target column in the AIF table where data will be written. The target table in AIF is called RSE_PR_LC_CAL_CDA. So RSE_MD_CDA has the mapping between FACT table/columns and RSE_PR_LC_CAL_CDA columns. To make the setup easier, the configuration of metadata is done programmatically. AIF checks for columns that have data for at least one week in any of the four FACT tables. Based on that, the code populates RSE_MD_CDA. It uses the FACT column names to determine the data type for that column. In RSE_MD_CDA, there is a column for UI_DISPLAY_NAME that the user can optionally edit, using the System Configuration under Control & Tactical Center.
- RSE_MD_FACT_COLUMN_MAP:** This table contains the mapping between columns in RSE_PR_LC_CAL_CDA and the final target tables such as sales and inventory in AIF. This table has four columns for SOURCE_TABLE, SOURCE_COLUMN, TARGET_TABLE, and TARGET_COLUMN. The source columns have the names of FACT tables/columns. The target columns have the names of tables in AIF that must be populated based on the FACT data. For example, if the FACT measures are supposed to be used in PMO/Forecasting, the mapping between the FACT table/column and the PMO_ACTIVITIES columns must be defined in RSE_MD_FACT_COLUMN_MAP. Another example is populating inventory and sales tables in AIF based on FACT measures. For a customer who wants to provide data at the aggregate level via a FACT measure, the mapping between the FACT tables and the sales/inventory tables in AIF must be defined in RSE_MD_FACT_COLUMN_MAP. It is important to insert the proper data for mapping into RSE_MD_FACT_COLUMN_MAP before loading data. The user can add mappings in RSE_MD_FACT_COLUMN_MAP, using the System Configurations Screen under Control & Tactical Center.

The following table provides some examples of configuring data in RSE_MD_FACT_COLUMN_MAP for the most common columns such as sales quantity and amount. These examples assume that the data is being provided via W_RTL_FACT1_PROD1_LC1_T1_F and that the intersection is SKU/store/week. The exact mapping will vary by each implementation and will depend on which FACT interface(s) and which columns are used and what measure they represent.

Table 17-9 Examples of Configuring Data

Domain	SOURCE_TABLE	SOURCE_COLUMN	TARGET_TABLE	TARGET_COLUMN
Sales	W_RTL_FACT1_PROD1_LC1_T1_F	SLSRG_QTY	RSE_SLS_PR_LC_WK	SLS_QTY
	W_RTL_FACT1_PROD1_LC1_T1_F	SLSRG_RTL_AMT	RSE_SLS_PR_LC_WK	SLS_AMT
	W_RTL_FACT1_PROD1_LC1_T1_F	SLSPR_QTY	RSE_SLS_PR_LC_WK	SLS_PR_QTY
	W_RTL_FACT1_PROD1_LC1_T1_F	SLSPR_RTL_AMT	RSE_SLS_PR_LC_WK	SLS_PR_AMT
	W_RTL_FACT1_PROD1_LC1_T1_F	SLSCL_QTY	RSE_SLS_PR_LC_WK	SLS_CLR_QTY
Inventory	W_RTL_FACT1_PROD1_LC1_T1_F	EOH_QTY	RSE_INV_PR_LC_WK	INV_QTY_BOH

POM Jobs

The following table shows the POM jobs that must be run to load data from FACT tables into AIF.

Table 17-10 POM Jobs

JobName	Description	RMSBatch	ProcessName
RSE_AGGREGATE_METADATA_LOAD_ADHOC_JOB	Ad hoc job to load aggregate measure metadata from RA to RSE	rse_aggregate_metadata_load.ksh	RSE_LOAD_AGGREGATE_FACT_DATA_ADHOC_PROCESS
RSE_AGGREGATE_ACTUALS_LOAD_ADHOC_JOB	Ad hoc job to load aggregate measure actuals from RA to RSE	rse_aggregate_actuals_load.ksh	RSE_LOAD_AGGREGATE_FACT_DATA_ADHOC_PROCESS
RSE_SLS_PR_LC_WK_AGGR_MEAS_SETUP_ADHOC_JOB	Ad hoc job for product at leaf level/location/week sales aggregate measure setup	rse_sls_pr_lc_wk_aggr_meas_setup.ksh	RSE_LOAD_AGGREGATE_FACT_DATA_ADHOC_PROCESS
RSE_SLS_PR_LC_WK_AGGR_MEAS_PROCESS_ADHOC_JOB	Ad hoc job for product at leaf level/location/week sales aggregate measure process	rse_sls_pr_lc_wk_aggr_meas_process.ksh	RSE_LOAD_AGGREGATE_FACT_DATA_ADHOC_PROCESS
RSE_SLS_PH_LC_WK_AGGR_MEAS_SETUP_ADHOC_JOB	Ad hoc job for product at non-leaf level/location/week sales aggregate measure setup	rse_sls_ph_lc_wk_aggr_meas_setup.ksh	RSE_LOAD_AGGREGATE_FACT_DATA_ADHOC_PROCESS
RSE_SLS_PH_LC_WK_AGGR_MEAS_PROCESS_ADHOC_JOB	Ad hoc job for product at non-leaf level/location/week sales aggregate measure process	rse_sls_ph_lc_wk_aggr_meas_process.ksh	RSE_LOAD_AGGREGATE_FACT_DATA_ADHOC_PROCESS

Table 17-10 (Cont.) POM Jobs

JobName	Description	RMSBatch	ProcessName
RSE_INV_PR_LC_WK_ AGGR_MEAS_SETUP_ ADHOC_JOB	Ad hoc job for product at leaf level/location/week inventory aggregate measure setup	rse_inv_pr_lc_wk_aggr_ meas_setup.ksh	RSE_LOAD_AGGREGA TE_FACT_DATA_ADHO C_PROCESS
RSE_INV_PR_LC_WK_ AGGR_MEAS_PROCES S_ADHOC_JOB	Ad hoc job for product at leaf level/location/week inventory aggregate measure process	rse_inv_pr_lc_wk_aggr_ meas_process.ksh	RSE_LOAD_AGGREGA TE_FACT_DATA_ADHO C_PROCESS

Internationalization

The user interface supports multiple languages in a single instance, but the underlying database only supports a single language in an instance.

The database default language is selected at installation. Once set, there is no support for switching the database language. The application user interfaces adhere to the language setting for each user's browser. For example, to change the language for the Firefox browser:

1. Select Tools from the menu bar.
2. Select Options.
3. Select Choose.
4. Select the language to add.

The following language are supported: English, German, Greek, Spanish, French, Croatian, Hungarian, Italian, Japanese, Korean, Dutch, Polish, Brazilian Portuguese, Russian, Swedish, Turkish, Simplified Chinese, and Traditional Chinese.

Configuration Updates

Use the Manage Configuration user interface screen in order to edit the values in the configuration tables. In order to access this screen, you must be assigned the role of ADMINISTRATOR_JOB. The tables you can edit within Manage Configuration depends upon which application roles you have been assigned, as described in [Table 17-1](#).

Once you have accessed Manage Configuration, you select, from the configuration tables available to you, the table that you want to review and edit. Note that each configuration table allows specific actions. You may or may not be able to view the table, edit some or all of the columns, or add and delete rows.

Certain of the tables in Manage Configuration are not intended for configuration but instead are provided to display data that may be useful during implementation. For each application interface file, a table is available that contains any records that have failed the data validation rules. Such tables have an extension of BAD. For example, for the rse_prod_attr_grp_value_stg.txt interface, the data is loaded into a table named RSE_PROD_ATTR_GRP_VALUE_STG. The error table is named RSE_PROD_ATTR_GRP_VALUE_BAD.

You can see all the error tables for all the interfaces via Manage Configuration. When an error occurs with an interface, you can review the BAD table to see what records failed validation and why. Each BAD table includes the following columns: ERROR_ROWID, ERROR_ID,

ERROR_DESCR, and ERROR_DT. The ERROR_ID is useful as a reference. The ERROR_DESCR provides a message to explain the validation error.

Configuration Tables

The following tables can be viewed in the Manage Configuration screen.

Table 17-11 Customer Decision Tree Configuration Tables

Table Name	Description
CDT_EXCLUDE	Defines the various types of pruning filters that can be used to prevent a CDT from being used during the escalation phase of the CDT workflow.
CDT_FILTER	Defines the various types of data filters that can be used to filter sales transaction data used for the CDT calculation.
CDT_VERSION	Defines a version to collectively group together a batch of CDTs that have been created for a particular purpose.

Table 17-12 Advanced Clustering and Customer Segmentation Configuration Tables

Table Name	Description
CIS_ALGORITHM_ATTR	Defines the possible attributes for any algorithm.
CIS_BUSINESS_OBJECT	Hosts the list of applications that are registered and configured to use the clustering feature.
CIS_BUSSOBJ_OBJ_ALG_XREF	This cross reference is provided so that you can use the same algorithm to generate different cluster objectives. The same algorithm can be used to generate customer clusters as well as store clusters. At the same time it is possible to list multiple algorithms that can be used to achieve a similar objective.
CIS_BUSSOBJ_TCRIT_HIER_XREF	Specifies the possible hierarchy levels for each hierarchy type (merchandise and location) that are permitted for the combination of objective ID, business objective ID, and type criteria ID.
CIS_BUS_OBJ_HIER_DEPLOY_XREF	Identifies the business object, objectives, product hierarchy types, and levels that can be selected for deployment. This ensures that only authorized types of clusters are exported to external systems (CMPO).
CIS_BUS_OBJ_NESTED_CRITERIA	Determines the possible child cluster types for a parent cluster.
CIS_BUS_OBJ_TCRITERIA_XREF	Specifies the possible cluster types that are permitted for the combination of objective ID and business objective ID.
CIS_BUS_OBJ_TCRIT_ALGO_ATTR	Defines possible attributes for any algorithm, business objective, objective, and criteria.
CIS_CLUSTER_OUTLIER_RULE	Specifies the possible outlier rules for a type of criteria.
CIS_EFFECTIVE_PERIOD	Specifies the planning period information.
CIS_ODM_SETTINGS	Temporary storage for the data mining settings provided to ODM. You can store multiple runs of settings by using different mining functions and run IDs. To provide this data to ODM, you must write a view that provides the appropriate rows of data.
CIS_SALES_TYPE	Specifies the type of sales information - historical, forecasted, or planned.

Table 17-12 (Cont.) Advanced Clustering and Customer Segmentation Configuration Tables

Table Name	Description
CIS_SRC_ENTITY	Defines the different database views available for use in the various clustering implementations. These settings help control how the attributes are used throughout the system.
CIS_TCRITERIA_SRC_XREF	Cross reference to SRC_ENTITY_NAME and for the settings for Partitioning, Information for the SRC_ENTITY_NAME attributes.
CIS_TCRIT_SRC_TYPE_XREF	Describes the xref of type_criteria and sales.
CIS_TYPE_CRITERIA	List of different clustering types and criteria that can be used to generate clusters.

Table 17-13 Demand Transference Configuration Tables

Table Name	Description
DT_EXCLUDE	Defines the different types of pruning filters available to prevent a DT result from being used during the escalation phase of the DT workflow.
DT_FILTER	Defines the different types of data filters used during the DT data filtering process.

Table 17-14 Generic Configuration Tables

Table Name	Description
RSE_AGGR_SRVC_CONFIG_LEVELS	Defines the different hierarchy types and levels that must have aggregated data created as part of the hierarchy configuration.
RSE_BUSINESS_OBJECT_ATTR_MD	Defines the attributes for business objects and also provides relevant details about where from external table sources to obtain the data for this attribute.
RSE_BUSINESS_OBJECT_DB_SRC	Defines the source database objects for the attributes.
RSE_CONFIG	Provides configuration names and their values for the settings that can be changed and affect the operation of the software.
RSE_CONFIG_CODE	Provides configuration values for those configurations that can have different values, depending on how other values are configured. For example, if a configuration is required for a default error tolerance, but department 1 requires a different value, then a row here with a PARAM_CODE of 1 will enable a different value than the base configuration in RSE_CONFIG for just that department.
RSE_EMAIL_CFG_DISTR	Intersection table between the email configuration table and the email distribution list table used to resolve the many-to-many relationships.
RSE_EMAIL_DISTR_LIST	Provides the email distribution lists.
RSE_EMAIL_NOTIF_CFG	Defines the email messaging configuration.
RSE_EXP_GRP	Hosts the list of applications that are registered and configured to use the clustering application.
RSE_HIER_LEVEL	Defines the various levels for all the hierarchies.
RSE_HIER_TYPE	Defines the hierarchies that are available for use by the RSE applications.

Table 17-14 (Cont.) Generic Configuration Tables

Table Name	Description
RSE_LOAD_SRVC_CFG	Defines a data loader that can be executed through the data loading framework.
RSE_LOAD_VALDT_RULES_CFG	Defines the validation rules that a data loader performs, along with some configurable options that impact rows that fail the validation.
RSE_PROC_TASK_TMPL	Defines the templates for processing tasks used by the applications.
RSE_PROC_TMPL	Defines the processing templates for asynchronous or synchronous invocable from Java applications.
RSE_SRVC_CONFIG	Defines all the database service routines that can be invoked through the database service framework in RSE.

Email Notification Configuration

The RSE_EMAIL_CFG_DISTR, RSE_EMAIL_DISTR_LIST, and RSE_EMAIL_NOTIF_CFG tables listed in the tables in [Configuration Tables](#) can be used to enable email notifications for certain batch processes. The RSE_EMAIL_NOTIF_CFG table contains a list of processes that can generate an email notification. Configurations are available for loading interface files (PROCESS_TYPE=LOAD), as well as configurations for running the automated batch processes (PROCESS_TYPE=BATCH).

The RSE_EMAIL_DISTR_LIST table is used to define email distribution lists that should receive email notifications. If all email notifications are to be sent to the same email address, then a row with a DEFAULT_FLG=Y can be used so that the distribution list is used for all email notices.

If some processes require different email recipients, then additional distribution lists can be created for those in RSE_EMAIL_DISTR_LIST. Once created there, a record can be created in RSE_EMAIL_CFG_DISTR that links the distribution list with the email notification. Any email configuration without a specific entry in RSE_EMAIL_CFG_DISTR is sent to the default distribution list.

By default, all LOAD failures trigger an email notification with details about the load that failed. Additionally, all batch processes send an email notification when the entire batch process completes. This email indicates whether the batch process was successful or if a failure occurred.

Attribute Processing

This chapter addresses attribute preprocessing. It contains the following sections:

- [Attribute Preprocessing](#)
- [Product Attribute Loading](#)

Attribute Preprocessing

Attributes provide context about products and enhance the accuracy of DT and CDT models. Attributes are stored within AIF data warehouse and are derived from product descriptions and merchandise hierarchy.

AIF applications may or may not contain product attributes. Any attributes found in AIF applications may have been created for BI reporting or other purposes and may need mining or preprocessing to make them suitable for the application.

Some steps in attribute preprocessing require manipulating attribute data. Oracle Enterprise Product Data Quality is a licensed software package that facilitates some of the preprocessing data manipulation steps required to make attributes suitable for CDT and DT modeling.

Here is an example of product information for yogurt.

- Product description: Brand A non-fat organic 6 oz.
- Class description: Dairy product.
- Sub-class description: Yogurt.

SKU/Store attributes determined by preprocessing:

- Brand
- Price
- Size

Note that CDT modeling works optimally when there are five or fewer possible values for any given SKU-store attribute. For example, many price points are available for yogurt. For CDT, it is better to define between three and five price bins (that is, budget, regular, premium, and elite).

Process Overview

The basic steps for attribute preprocessing are as follows:

- Populating AIF data warehouse with attribute data
- Translating (optional)
- Parsing
- Cleansing and standardizing
- Categorizing and labeling
- Defining attributes

- Binning and grouping

Populating AIF Data Warehouse with Attribute Data

A few steps are required to make attributes suitable for the application so that the applications can use this data.

The first requirement is to ensure that the attribute values are populated in the AIF data warehouse (W_RTL_ITEM_GRP1_D table). This is the source for the attribute data and must be loaded there in order to be available to the application.

Regarding raw attributes: In the data warehouse, an attribute can be defined in multiple ways. Some generic attributes are typically stored in a column of the W_PRODUCT_ATTR_D table. The data warehouse has a table W_RTL_METADATA_G that contains a list of defined attribute locations. Consult this list to see if there is already a defined place to store a particular attribute value.

AIF data warehouse also offers the ability to store Item Differentiators for products. These are essentially User Defined Attributes (UDAs), which consist of a lookup code for the attribute and the attribute value. These lookup codes are then defined in a standard translation table (W_DOMAIN_MEMBER_LKP_TL with domain codes of ITEM_UDA_HEAD and ITEM_UDA). The actual association of an item to one of the UDAs is performed in the W_RTL_ITEM_GRP1_D table.

Once attributes are available in the data warehouse, it is necessary to define these attributes in the RSE_BUSINESS_OBJECT_ATTR_MD table. This table must be set up with appropriate metadata to define the source of the attributes from the data warehouse tables. The sample seed_data file for this table contains some standard attributes that can be defined in W_RTL_ITEM_GRP1_D, but the table must be adjusted to contain the complete list of attributes that should be available for the applications to use. This must include Flex Attributes as well as User Defined Attributes. Once attributes are defined in the RSE_BUSINESS_OBJECT_ATTR_MD, the next step is to provide custom lists of attributes that should be used per product category. This can be done through the RSE_PROD_ATTR_GRP_VALUE_STG and RSE_PROD_ATTR_VALUE_XREF_STG interfaces. The first interface is used to define the output of the binning and grouping of attributes. For example, if Coffee needs a Brand Tier attribute, and it should have values of Premium, Value, and Mainstream, then this interface would define this Coffee Brand Tier attribute, along with the values of Premium, Value, and Mainstream, and it should specify what source attribute is to be used for this (the source is in RSE_BUSINESS_OBJECT_ATTR_MD). The second table of the interface (RSE_PROD_ATTR_VALUE_XREF_STG), enables the association of specific Brand attributes to the binned/grouped attribute values from the first interface (RSE_PROD_ATTR_GRP_VALUE_STG). One concept to consider for these attributes and attribute values, is that they must be unique across all product categories. This offers the ability to classify one Brand as Premium for one product category, while it could be Mainstream for another product category. Additionally, it enables a different selection of attribute values for each product category. For example, another product category might not have a Premium Brand Tier, and therefore the interface would not include this value in this interface for that product category.

Translating

This step is only needed when the product data is in a different language than the customer's primary language.

Parsing

This step identifies and extracts target key words, such as "Brand A," "small," "blue," and "non-fat." from the source data (such as product description). It is done through semantic recognition, usually by software such as Oracle Enterprise Product Data Quality.

Cleansing and Standardization

This step edits the text and corrects spelling and grammar. For example, "Addr." will be recognized and converted into "Address" and "St." into "Street." EPDQ can facilitate this step.

Categorizing and Labeling

This step classifies targeted key words into the pre-defined categories, such as "Brand A" for "Brand," "small" for "Size" and "blue" for "Color." The product record can thus be labeled by the category values. EPDQ can facilitate this step.

Defining Attributes

With the extracted categories from the product data, attributes are defined. They can be some or all of the categories identified, based on contextual business knowledge and how populated the categories are.

Binning and Grouping

Binning and grouping are used to consolidate and reduce the number of possible values for an attribute into a manageable number.

- Binning divides numerical attributes, such as price, discounts, and mileage, into discrete sets of ranges, such as <=\$10, \$10~\$25, and >\$25.
- Grouping combines textual attributes that are too granular into a smaller set of attribute values. For example, tea weight can have dozens of values; grouping merges the values into coarser ranges (like small or large) and reduces the number of possible attribute values.

Product Attribute Loading

This section provides an example of adding an attribute for use by the application into all the relevant tables. In this example, a new attribute is added to represent Flavor within the Coffee product category.

The process flow for this involves:

1. Identify the need to add a new product attribute for a product category.
2. Determine where the attribute data is found within the data warehouse.
3. Use the Manage System Configurations screen to add the attribute definition in the tables, if it not already present.
4. Use AIF Apps jobs in POM to run the batch process to load attribute data from the data warehouse.
5. Determine if the attribute data requires any special grouping or binning.
6. Populate the RSE_PROD_ATTR_GRP_VALUE_STG staging table with attribute definition and values.
7. Populate the RSE_PROD_ATTR_VALUE_XREF_STG staging table with data to associate raw RADM attribute values to the Attribute Groups defined above.

8. Use AIF Apps jobs in POM to run the batch process that processes the interface staging tables.
9. Use AIF Apps jobs in POM to update the CIS attribute data to reflect the new attribute (product attributes).
10. Use AIF Apps jobs in POM to update the CIS attribute data to reflect the new attributes (non-product attributes).

Introduce New Attribute

The first step in the process is the catalyst that triggers the remaining steps. The catalyst is the new attribute that has been introduced and must be made available within the application.

Determine the Attribute Source and Define in the Tables

The new attribute is loaded from the AIF data warehouse for each of the products that require this attribute. The data warehouse has multiple ways of loading attributes, so the approach used varies, depending on where and how the data is stored. The process involves defining the source table and then defining the column (or column filter values) used to identify the attribute. Once the source is determined, the appropriate values are loaded into RSE_BUSINESS_OBJECT_ATTR_MD and possibly RSE_BUSINESS_OBJECT_DB_SRC.

W_PRODUCT_D or W_PRODUCT_ATTR_D

The data Warehouse's W_PRODUCT_D table and W_PRODUCT_ATTR_D table can provide attributes from any of the available columns in these tables. The W_PRODUCT_D table contains named columns with data of a specific logical value, while the W_PRODUCT_ATTR_D table contains a more flexible set of Number, Text, and Date columns that can contain varying values, depending on the implementation. From an attribute point of view, these tables are effectively the same and require the same type of handling.

W_RTL_ITEM_GRP1_D

The W_RTL_ITEM_GRP1_D table in the data warehouse is different than the other product attribute sources, in that these tables can have attributes implemented as unique rows and specific columns. This table contains a PROD_GRP_TYPE column, which defines the type of data in the table. Values of ITEMUDA are used for User Defined Attributes. Rows in which the PROD_GRP_TYPE corresponds to the BRAND, COLOR, FLAVOR, SCENT, FABRIC, and STYLE columns (ex. BRAND_WID) are also possible.

Populate RSE_PROD_ATTR_GRP_VALUE_STG Interface

Once the attribute data has been reviewed and groups have been defined, it is necessary to define the attribute groups and process them into the database. The output of the prior step must be loaded into the staging table for Attribute Value Groups (RSE_PROD_ATTR_GRP_VALUE_STG). This interface defines two sets of data and is used to load two different tables.

Table 18-1 RSE_PROD_ATTR_GRP_VALUE_STG

Column	Example	Description
PROD_HIER_TYPE_NAME	Product Hierarchy	Must match the NAME from RSE_HIER_TYPE that has the ID equal to the RSE_CONFIG for CMGRP_HIER_TYPE.
PROD_EXT_KEY	CLS~1000~10000	The external key used to identify the product category (for example, Coffee Class). This value is the same as in RADM's INTEGRATION_ID of the W_PROD_CAT_DH, and also the PROD_EXT_KEY of the RSE_PROD_SRC_XREF table.
ATTR_SHORT_DB_NAME	FLAVOR	This must match the SHORT_DB_NAME of the RSE_BUSINESS_OBJECT_ATTR_MD table for the newly added attribute.
PROD_ATTR_GRP_EXT_KEY	CLS~1000~10000~flavor_yn CLS~1000~10000~flavor_type	This must be a unique value to describe the attribute to be used by the applications. Since the source Flavor attribute is being defined as two different attributes, two example values are shown here.
PROD_ATTR_GRP_NAME	FlavorYN FlavorType	A name to be displayed in the UI for the new attribute. Two example values are shown here.
PROD_ATTR_GRP_DESCR	Flavor Y/N Identifier Flavor Type	An optional/additional descriptive value that can be displayed in the UI for the new attribute.
PROD_ATTR_VALUE_KEY	(See additional table below)	A unique/external identifier for the new attribute values.
PROD_ATTR_VALUE_NAME	(See additional table below)	A name displayed in the UI for the attribute value.
PROD_ATTR_VALUE_DESCR	(See additional table below)	An optional/additional descriptive value that could be shown in the UI for the new attribute value.
FUNC_ATTR_FLG	N	This is a Y/N flag to indicate whether this attribute is considered to be an attribute associated with a specific function or role (Y) or not (N). For example, a customer cannot choose a product with a different value for the auto wiper blade size because each car model has a specific size requirements.

Here is a table showing the different values for adding the example Flavor Attribute Values.

Table 18-2 Flavor Attribute Values

PROD_ATTR_GRP_NAME	PROD_ATTR_VALUE_KEY	PROD_ATTR_VALUE_NAME	PROD_ATTR_VALUE_DESCR
FlavorYN	CLS~1000~10000~flavor_yn~y	Y	Yes
FlavorYN	CLS~1000~10000~flavor_yn~n	N	No
FlavorType	CLS~1000~10000~flavor_type~non	Non Flavored	Non Flavored

Table 18-2 (Cont.) Flavor Attribute Values

PROD_ATTR_GRP_NAME	PROD_ATTR_VALUE_KEY	PROD_ATTR_VALUE_NAME	PROD_ATTR_VALUE_DESCR
FlavorType	CLS~1000~10000~flavor_type~fruit	Fruit Flavored	Fruit Flavored
FlavorType	CLS~1000~10000~flavor_type~mild	Mild Flavored	Mild Flavored
FlavorType	CLS~1000~10000~flavor_type~special	Specialty	Specialty

Populate RSE_PROD_ATTR_VALUE_XREF_STG Interface

Once the RSE_PROD_ATTR_GRP_VALUE_STG interface has been loaded, it is possible to load the RSE_PROD_ATTR_VALUE_XREF_STG interface with a mapping of actual product attribute values (otherwise known as base attributes) to the attribute groups that were loaded via RSE_PROD_ATTR_GRP_VALUE_STG. The format of data to be loaded here depends on the format of the base attributes. Only one set of attribute value columns should be populated for this interface. These sets are MIN_ATTR_NUM_VALUE and MAX_ATTR_NUM_VALUE (for numeric attributes), ATTR_STRING_VALUE (for text attributes), MIN_ATTR_DATE_VALUE and MAX_ATTR_DATE_VALUE (for date attributes), ATTR_VALUE_EXT_CODE (for dimension-based attributes). The sets are mutually exclusive of each other for this interface.

Table 18-3 RSE_PROD_ATTR_VALUE_XREF_STG

Column	Example	Description
PROD_ATTR_VALUE_KEY_Y	CLS~1000~10000~flavor_type~y	Must match a PROD_ATTR_VALUE_KEY that was loaded via the RSE_PROD_ATTR_GRP_VALUE_STG interface.
MIN_ATTR_NUM_VALUE	0	Minimum numeric value to associate with this attribute group value. Only applicable if this attribute uses the ATTR_NUM_VALUE column to store the base attribute value.
MAX_ATTR_NUM_VALUE	7	The maximum numeric value to associate with this range. Only applicable in conjunction with MIN_ATTR_NUM_VALUE.
ATTR_STRING_VALUE	Y	A string value to associate with this attribute group value. Only applicable if this attribute uses the ATTR_STRING_VALUE column to store the base attribute value.
MIN_ATTR_DATE_VALUE	2010-01-01	The minimum date value to associate with this attribute group value. Default date format for provided control file is YYYY-MM-DD. Only applicable if this attribute uses the ATTR_DATE_VALUE column to store the base attribute value.
MAX_ATTR_DATE_VALUE	2010-01-31	The maximum date value to associate with this attribute group value. Default date format for provided control file is YYYY-MM-DD. Only applicable in conjunction with MIN_ATTR_DATE_VALUE.

Table 18-3 (Cont.) RSE_PROD_ATTR_VALUE_XREF_STG

Column	Example	Description
ATTR_VALUE_EXT_CODE	32E	For base attributes that are sourced from W_RTL_ITEM_GRP1_D, this column can be used to specify the key from the appropriate source column. This is applicable if this attribute uses ATTR_VALUE_EXT_CODE to store the attribute value.

Here is a table containing some examples for adding a new flavor attribute, using string-based attributes.

Table 18-4 Adding a New Flavor Attribute

PROD_ATTR_VALUE_KEY	ATTR_STRING_VALUE
CLS~1000~10000~flavor_yn~y	BLUEBERRY
CLS~1000~10000~flavor_yn~y	RASPBERRY
CLS~1000~10000~flavor_yn~y	VANILLA
S~1000~10000~flavor_yn~y	CARAMEL
CLS~1000~10000~flavor_yn~y	CINNAMON
CLS~1000~10000~flavor_yn~y	HAZELNUT
CLS~1000~10000~flavor_yn~n	PLAIN
CLS~1000~10000~flavor_type~non	PLAIN
CLS~1000~10000~flavor_type~fruit	BLUEBERRY
CLS~1000~10000~flavor_type~fruit	RASPBERRY
CLS~1000~10000~flavor_type~mild	HAZELNUT
CLS~1000~10000~flavor_type~mild	VANILLA
CLS~1000~10000~flavor_type~special	CINNAMON
CLS~1000~10000~flavor_type~special	CARAMEL

AI Foundation Web Services

This chapter provides an overview of the AI Foundation Web Services. For additional details about any interfaces, see *Oracle Retail Insights Cloud Service Suite/Oracle Retail Analytics and Planning Cloud Services Data Interface*.

AI Foundation Web Services

AI Foundation web services provide access to DB-bound configurations when a direct connection to the database is not desirable or possible; intra-day and ad-hoc access to certain application outputs is also provided. AI Foundation web services are REST-based; it is assumed that you are familiar with basic REST principles (such as the usage of HTTP verbs). AI Foundation web services provide access to a subset of application and output data, but do not fully mirror the user interface or export and import features of the backend. They are not a replacement for bulk data export, which must be done on a schedule as part of batch processing. However, access to the configuration can be used during implementation and upgrade periods, and the AC export web service can serve as a means of obtaining incremental update data from a specified point in time (driven by a query parameter) as a means of intra-day processing.

All services support the query parameter `contentType` and the HTTP header `Content-Type`, with the supported values `application/json` and `application/xml`. The query parameter takes precedence; if no content type is supplied, then `application/json` serves as the default.

The AC export service has a `dateMask` query parameter that must adhere to the Java `java.text.SimpleDateFormat` rules (for example, `dateMask=yyyyMMdd&exportDate=20151012`). All date parameters must be sent in this format, and output dates and timestamps are returned according to this format.

The json/XML structure follows the corresponding DB table and view while being converted to the Java standard. That is, underscores are removed, camel case is used, and first letter is lowercase. For example, the `RSE_CONFIG APPL_CODE` column is returned as `applCode`. All data is returned as type string.

Authentication and Authorization

Basic authentication is used, so you may use any client software that supports it. Authorization is done for ADF-LDAP (OID) mapped roles, and only administrator roles are used (that is, the calling user must be in a duty that is mapped to the roles in [Table 19-1](#)).

Table 19-1 Mapped Administrator Roles

Service	Role	Mapped Role
DT	DemandTransferenceRole	ANALYTIC_EXPERT_JOB
CDT	CustomerDecisionTreeRole	ANALYTIC_EXPERT_JOB
AC	StoreClusterAdvancedRole	CLUSTERING_ADMINISTRATOR_JOB
CS	CustomerSegmentAdvancedRole	CUSTOMER_SEGMENT_ADMINISTRATOR_JOB

Summary of Web Services

This section provides a summary of web services.

Access to RSE_CONFIG Table

Fetch Config Data

GET on `/rase/resources/rse/parameters` returns all RSE_CONFIG entries a specific user has access to (that is, all applCode RSE entries plus corresponding application entries). For example, if a user is in an Administrator role, then the applCode SO will be returned as well. Here is a list of the fields:

- applCode
- paramName
- paramValue
- configurableFlg.
- efere

Fetch One Entry

GET on `applCode,paramName`. For example, `/rase/resources/rse/parameters/RSE,PRIMARY_LANGUAGE_CODE` returns a particular entry if the user has access to it.

Create an Entry

POST to `/rase/resources/rse/parameters` with a form having a single field called "content" and a value having the appropriate new parameter data in content type as specified by query parameter and header, if the user has access to it. The structure of the content must match what is returned by GET.

Update an Entry

PUT to `applCode,paramName`. For example, `/rase/resources/rse/parameters/RSE,LOC_HIER_TYPE`, with the form having fields called `paramName`, `paramValue`, `efere`, and `configurableFlg`, along with appropriate values, if the user has access to it.

Batch Update Entries

PUT to `/rase/resources/rse/parameters` with a form having a single field called "content" and a value having appropriate parameter data in the content type as specified by the query parameter and header. Note that if the user does not have access to a particular entry, it will be skipped. The structure of the content must match what is returned by GET.

Access to RSE_CONFIG_CODE Table

Fetch Data for an Entry

GET on `applCode,paramName, paramCode`. For example, `/rase/resources/rse/parameters/DT,SIM_DISPLAY_CODE_PCT,2` returns data, if user has access to it.

- applCode

- paramName
- paramCode
- paramValue
- configurableFlg
- efere

Create an Entry

POST to applCode,paramName,paramCode. For example, /rase/resources/rse/parameters/DT,SIM_DISPLAY_CODE_PCT,2} with a form having a single field named "content" and a value having appropriate new parameter data in the content type as specified by query parameter and header, if the user has access to it. The structure of the content must match what is returned by GET.

Update an Entry

PUT to applCode,paramName, paramCode, For example, /rase/resources/rse/parameters/DT,SIM_DISPLAY_CODE_PCT,2 with form having a single field named "content" and a value having appropriate parameter data in the content type as specified by query parameter / header, if user has access to it. The structure of the content must match what is returned by GET.

Delete an Entry

DELETE to applCode,paramName, paramCode. For example, /rase/resources/rse/parameters/DT,SIM_DISPLAY_CODE_PCT,2, if the user has access to it.

Advanced Clustering Export

This section describes Advanced Clustering export.

Get Clusters

This service is based on rsestrclst.csv.

GET on /rase/resources/cis/export/cluster.

Parameters (in addition to dateMask and contentType) are all joined by logical AND:

Table 19-2 Advanced Clustering Export Parameters

Name	gType	Required	Query Logic for Corresponding Column if Parameter is Provided
exportedDt	Date	Required	Greater than or equal to this value
effStartDt	Date	Required	Greater than or equal to this value
effEndDt	Date	Required	Less than or equal to this value
prodHierTypeExtKey	String	Required	Equal to this value []
prodExtKey	String	Required	Equal to this value
locExtKey	String	Required	Equal to this value
locHierTypeExtKey	String	Required	Equal to this value

Customer Segment Export

This section describes Customer Segment export.

Get Segments

This service is based on rsestrclst.csv.

GET on /rase/resources/cis/export/segment.

Parameters (in addition to dateMask and contentType) are all joined by logical AND:

Table 19-3 Customer Segment Export Parameters

Name	Type	Query Logic for Corresponding Column if Parameter is Provided
exportedDt	Date	Greater than or equal to this value
effStartDt	Date	Greater than or equal to this value
effEndDt	Date	Less than or equal to this value
prodExtKey	String	Equal to this value

AIF Integration with ORCE (Customer Engagement)

This integration enables the application to send approved customer segments into Oracle Retail Customer Engagement (ORCE). Whenever customer segments are approved for merchandise, location in the application, a web service call is made to ORCE in order to save segments, along with its members and relevant attributes.

Credential Store

You can use the Manage Credential Stores screen, shown in [Figure 19-1](#), to maintain ORCE credentials in the application.

Figure 19-1 Manage Credential Stores

Enter the following information into Manage Credential Stores:

Table 19-4 Manage Credential Stores Information

Field	Description
Username	Credential username created by ORCE to enable integration
Password	Credential password (at least one character) created by ORCE to enable integration
Confirm Password	Confirm password
Description	Credential store used for Customer Segment integration with Oracle Customer Engagement

Configuration

The following configuration must be updated for integration. By default, ORCE integration is disabled.

Table 19-5 Configuration

Configuration	Description
CUST_SEG_WS_RELATE_FLG	Flag to identify whether to publish customer segment to customer engagement. (Y/N)
CUST_SEG_WS_RELATE_ORGID	ORGID is the 3-letter ID provided by Oracle Customer Engagement to create and implement an authentication key. (REL)
CUST_SEG_ATTR_WS_RELATE_FLG	Flag to identify whether to publish customer segment attribute to customer engagement. (Y/N)

Table 19-5 (Cont.) Configuration

Configuration	Description
CUST_SEG_RELATE_HOSTNAME	URL host name to publish customer segment to customer engagement. This should be same as the CN name in the certificate.
CUST_SEG_RELATE_PORT	Port number to publish customer segment to customer engagement.
CUST_SEG_REL_SEGSERV_VERSION	Segment service version to use for publishing customer segment to customer engagement. (format vX_0 - v3_0)
CUST_SEG_REL_PROXY_HOSTNAME	Proxy host name to publish customer segment to customer engagement.
CUST_SEG_REL_HTTP_PROXY_PORT	HTTP proxy port number to publish customer segment to customer engagement.
CUST_SEG_REL_HTTPS_PROXY_PORT	HTTPS proxy port number to publish customer segment to customer engagement.
CUST_SEG_REL_BATCH_SIZE	Number of customers to add in batches after customer segment to customer engagement is published.

Retail Science Integration with XStore

Retail Science integrates with XStore; as part of this integration, XCenter broadcasts real time sales data with multiple transactions in one XML file to Retail Science. Retail Science, on receiving data, saves real time sales transactions, any price modifiers like promotions, price overrides, and returns in the database, which can be used as real time sales data for reporting.

This section provides the information necessary to send messages contained in version 1.0 of the POSLOG Services API. Information can be obtained using the Web Service Description Language (WSDL) in conjunction with a Simple Object Access Protocol (SOAP), XML Schema, and various methods contained in the classes of the API to provide the web service described below.

Endpoint

The URL for the WSDL is:

`https://<host>:<port>/rasews/poslog/v1/PoslogStrReceiverApiService?wsdl`, where <host> is the name or address of the server. For a cloud installation, <port> the default port number is 443.

The URL for the WSDL is: Retail Science SOAP endpoint is: `https://<host>:<port>/rasews/poslog/v1/PoslogStrReceiverApiService` This endpoint should be configured in the XCenter for Generic Poslog String broadcaster. For further details, see the XCenter configuration.

Authentication

Retail Science supports Basic Authentication integration with XCenter over https, using a username and password. XCenter integration with Retail Science relies on SSL to do the encryption of the username/password configuration. After a user is defined in Retail Science, XCenter must be configured with the userid/password, and Xcenter will use Base64-encoding and include it as part of the Http authorization header when invoking the SOAP service.

XCenter service requests must be authenticated with a valid user/password combination and roles. Failure to provide a valid authorization header will result in a 403 - Forbidden response. The user must have following role set up: POSLOGS_SERVICE_JOB - Point of Sales broadcast listener role to enable integration between Retail Science and Oracle XStore.

Retail Science and Retail Insight Integration

Retail Science internally triggers job LOAD_POSLOG_DATA_ADHOC; this job is predefined in Retail Process Orchestration and Monitoring (POM). As part of the configuration verification, ensure that LOAD_POSLOG_DATA_ADHOC is defined in POM.

Configuration

This section describes the configuration.

POM Server Detail Configuration

To ensure AIF application to Retail Insights integration configuration is correct, verify RSE_CONFIG configuration for POM Server details. Select Control & Tactical Center > Strategy & Policy Management Filter APPL_CODE as "RSE" and PARAM_NAME as "RSE_POM_HOST". Determine if the value set as part of the deployment is correct. If the host is not correct, update the entries.

Figure 19-2 Edit Value

Edit Value	
APPL_CODE *	RSE
PARAM_NAME *	RSE_POM_HOST
PARAM_VALUE	https://glbu-phw-lbest-58.us.oracle.com
CONFIGURABLE_FLG *	Y
DESCR	POM Server Host for Retail Science application
UPDATEABLE_FLG	Y

Ok Cancel

POM Server Credentials

Select Control & Tactical Center > Manage Credential Store to provide POM ReST endpoints credentials for Basic Authentication. All the POM ReST endpoints use Basic Authentication, with the same credentials as those used to log into the POM application. See [Figure 19-3](#) to provide credentials.

Figure 19-3 POM Server Credentials

The screenshot shows the 'Manage Credential Stores' configuration page. On the left is a sidebar with icons for Recommendations, Manage Credential Stores (active), a star, a document, a bell with a red 'X', a bar chart, and a gear. The main content area has a breadcrumb 'Recommendations > Manage Credential Stores'. Below this are three dropdown menus: 'Select Credential Store' (Retail Science POM Credential Access), 'Select Credential Store Map' (oracle retail ris common reservations pom credentials), and 'Select Credential Store Key' (oracle retail ris common reservations pom credentials key). Below these are four input fields: 'Username', 'Password', 'Confirm Password', and 'Description'. A 'Create' button is at the bottom right of the form.

Once the configuration is complete, AIF applications will start triggering jobs for Retail Insights.

20

Batch Processing

This chapter provides an overview of the batch processing capabilities available for the application.

Overview

The implementation process involves loading data files for dimensions and fact data into the database. For new implementations, the best practice is to test the interfaces in a logical sequence, in small test cycles..

Once all required data has been loaded and all interfaces have been tested, the scheduled batch cycles that perform different tasks can be used, depending on the frequency involved. The application has DAILY, WEEKLY, and QUARTERLY batch cycles, each of which performs different tasks, depending on which applications are being used.

Incremental Exports

All incremental export files are controlled by a set of dates that define the beginning and ending range of data to be exported. This data is stored in a configuration table called RSE_EXP_GRP and can be seen in the Manage Configuration screen. Each incremental export has a date associated with the data to be exported. Only data that has a date/time value between the FROM_DT and TO_DT columns of the RSE_EXP_GRP that it is associated with, will be exported when the export file is created.

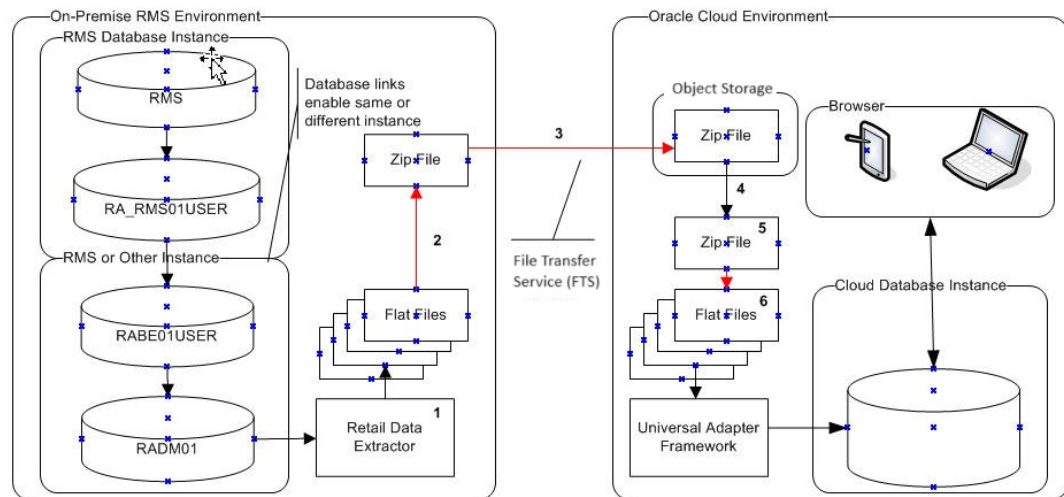
When testing an application, it is important to realize that if a test export of data is required, you must make sure that data is available to be exported and that the data is associated with a date that is in the range of the export group. If an export runs and does not produce any data in the file, you should check the values of the Export Group to ensure the dates were not set incorrectly.

When you create or process data in the application user interface, want to test the export of that data, you must advance Export Group's date range by running the appropriate export preparation step as described in this chapter. This causes the date range to advance and enables the exporting of the data that is available for exporting. Note that if the Export Group date range is advanced too many times, the data that you want to export may no longer be in the current range for exporting.

You may encounter such issues when using this custom batch process to trigger the execution of exports; however, these issues will not occur once the application is running the batch routines in an automated manner, because the batch processes are only executed once per batch cycle.

Batch Process Flow

[Figure 20-1](#) illustrates the batch process flow.

Figure 20-1 Batch Process Flow

Here is the process.

1. The on-premise batch shell script extracts data to files initiated by the customer scheduler.
2. Merch batch script creates the zip file named RI_RMS_DATA.zip. Additionally, zip files named RI_CE_DATA.zip and RI_MFP_DATA.zip should be created.
3. You should upload the three zip files to Object Storage using File Transfer Service (FTS).
4. Once the file is in Object Storage, the cloud service batch processing will retrieve the files from Object Storage after they have passed an anti-virus and malware scan.
5. The batch load process begins with tasks that
 - a. Archive the files that have been received in a date/time stamped directory.
 - b. Perform the presence validation exercise that verifies that all expected files for the customer's subscribed applications in the zipped files. This terminates if any expected files are missing.
 - c. Clear the previous day's files from the \$MMHOME/data/staging directory.
 - d. Unzip the zip file into the \$MMHOME/data/staging directory.

[Table 20-1](#) lists the zip files.

Table 20-1 Supported Zip Files

Zip File Name	Frequency	File Type	Notes
RI_RMS_DATA.zip	Daily	Inbound	All files which start with W_* can be placed in any combination of the RI*zip files.
RI_CE_DATA.zip	Daily	Inbound	All files which start with W_* can be placed in any combination of the RI*zip files.
RI_MFP_DATA.zip	Daily	Inbound	All files which start with W_* can be placed in any combination of the RI*zip files.
ORASE_WEEKLY.zip	Weekly	Inbound	Any inbound file that does not start with W_* and has a weekly frequency can be placed in here.

Table 20-1 (Cont.) Supported Zip Files

Zip File Name	Frequency	File Type	Notes
ORASE_INTRADAY.zip	Intraday	Inbound	Any inbound file that has an intraday frequency can be placed in here.
ORASE_WEEKLY_ADHOC.zip	Weekly	Inbound	Used for instances where AdHoc jobs are run. Any inbound file for that AdHoc job that does not start with W_* and has a weekly frequency can be placed in here.
ORASE_WEEKLY_extract.zip	Weekly	Outbound	Any outbound file that has a weekly frequency will be placed in here.
ORASE_INTRADAY_extract.zip	Intraday	Outbound	Any outbound file that has an intraday frequency will be placed in here.

Configuring Additional Data Files

It may be necessary to configure support for additional data files into AIF, beyond the ORASE_WEEKLY.zip and ORASE_INTRADAY.zip. If this is required, it is possible to configure additional files to go along with those. This section describes how to add support for this.

It is possible to receive additional zip files, with additional suffixes to the existing zip. In this example, the assumption is that a new zip file named ORASE_WEEKLY_IP.zip must be processed when ORASE_WEEKLY.zip is processed. The IP portion of this is what can be configured, as explained below.

Using the Retail AI Foundation Platform's UI, select the Data Management / Manage Configuration menu options. Then, select RSE_CONFIG_CODE as the table to configure. If you want to search for existing configurations, you can enter a search value of "%ZIP" in the PARAM_NAME Search field and select Search. There are no default extensions defined here, but once one has been created, this will display them.

To add a new entry to allow processing this additional file, select the Create Param icon to add a new configuration. You can enter values such as those shown in [Table 20-2](#) into the dialog box.

Table 20-2 Additional Data Files

Field	Value	Notes
APPL_CODE	RSE	
PARAM_NAME	ORASE_WEEKLY_ZIP	The format of this is important. If adding an additional file for the Intraday batch, the value would be ORASE_INTRADAY_ZIP instead.
PARAM_CODE	IP	This is the suffix that the additional zip will use.
PARAM_VALUE	Y	Y to enable this, or N to disable this file.
CONFIGURABLE_FLG	Y	Fixed value.
UPDATEABLE_FLG	N	Fixed value.
DESCR	Additional zip for IP files.	Adjust this description so it describes the zip contents/source.

Based on the example values in [Table 20-2](#), a new zip file named ORASE_WEEKLY_IP.zip will now be expected when the ORASE_WEEKLY.zip is expected.

If a file configured as defined above must be temporarily disabled, you can edit the PARAM_VALUE via the UI, so that it has a value of "N" instead of a value of "Y".

File Transmissions

Files required for processing such as the ORASE_WEEKLY.zip file and the ORASE_WEEKLY_IW.zip file must be uploaded to Object Storage using the File Transfer Service (FTS). For uploading to Object Storage, it must include the incoming prefix and file name. For more information on FTS, refer to the "Implementation Tools File Transfer Services" chapter in the *Oracle Retail Analytics and Planning Implementation Guide*.

Innovation Workbench

Innovation Workbench (IW) is a workspace that provides read-only access to application data objects and clean data by using Oracle APEX. This extension is a workspace for advanced analytics users to add new implementations by using Oracle Advanced Analytics (ODM) algorithms that are implemented as SQL/PLSQL functions. This chapter provides features, examples, and implementation details that are available in Innovation Workbench.

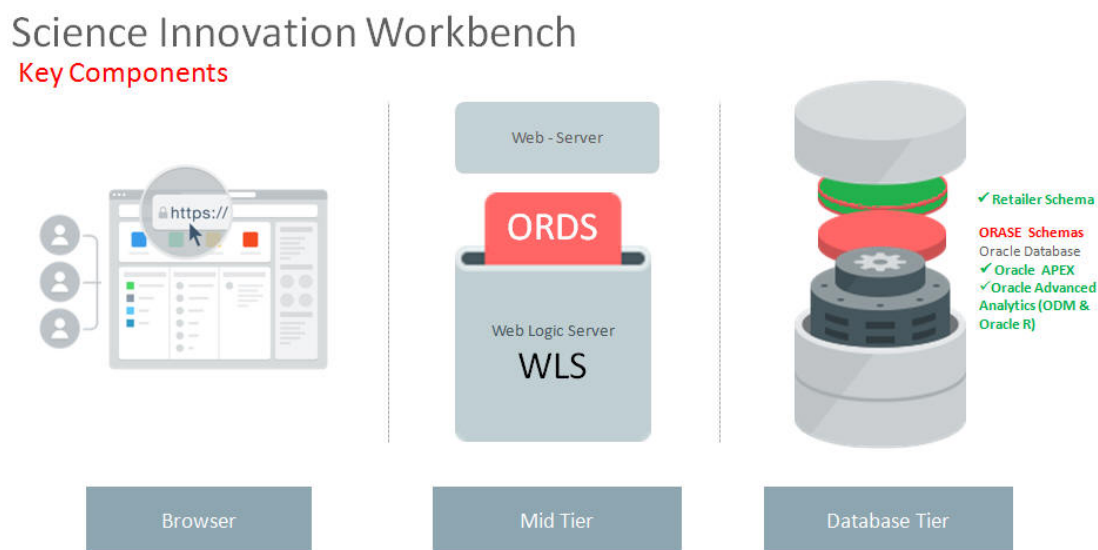
The key features available in Innovation Workbench are:

- Schema as a Service, in which a user can use AIF cleansed, read-only data, upload and combine data to gain insights, and upload retail application data to further analyze and mine data.
- Advanced Analytics, allows a user to use Oracle Machine Learning and Advanced Analytics algorithm that are implemented as SQL/PLSQL functions.
- Visualize Data, enables a user to explore data, develop and visualize data using charts, and review reports using the browser.
- RESTful Web Service, allows a user to declaratively create and edit RESTful service definitions using SQL Query.
- SQLWorkshop, enables a user to view and manage database objects.

Components

The key components of Science Innovation Workbench are Retailer Workspace Schema, Oracle APEX, and Oracle Advanced Analytics (Oracle Data Mining).

Figure 21-1 Innovation Workbench Key Components



Retailer Workspace Schema

Innovation Workbench provides retailer with a logical work area (Retailer Workspace) that is associated with a predefined database schema. The schema(s) store the database objects and provide read access to an existing retailer's application data objects. The retailer workspace schema is an independent schema for a retailer to use and these data objects are owned by retailer.

Oracle APEX

This section describes Oracle APEX, a web-based software development environment.

Workspace

A workspace <RETAILER_WORKSPACE> is a predefined workspace for the retailer where workspace users can create database objects and applications. The workspace has privileges to the allocated <RETAILER_WORKSPACE_SCHEMA> database schema.

Users and Roles

Innovation Workbench has two types of users: application developers and workspace administrators, and they can be created using Oracle APEX.

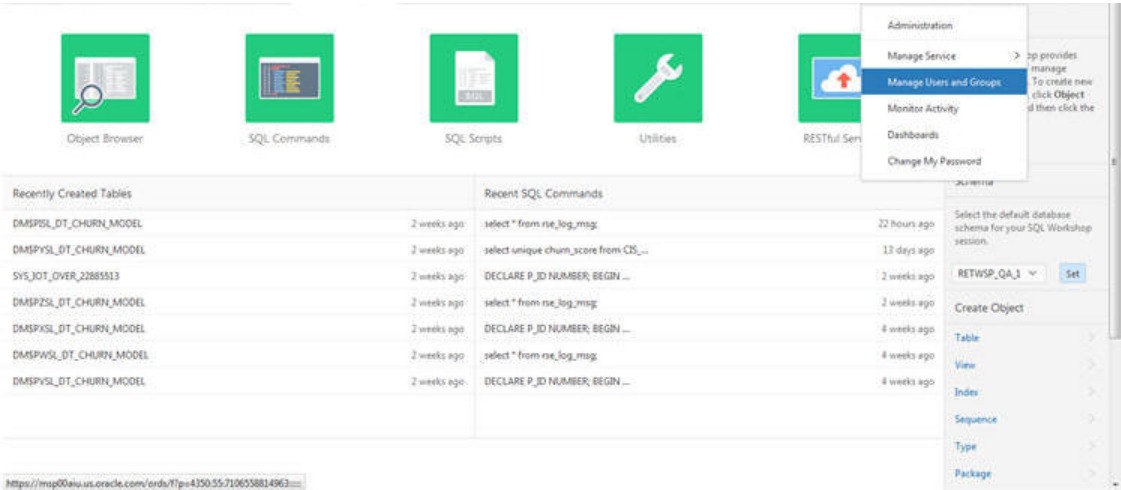
Innovation Workbench Administrators

The workspace administrator role is already created for the retailer; the administrator can create and edit developer accounts, manage groups, and manage development services.

Innovation Workbench Developer

Workspace administrators can create workbench developers by selecting Manage Users and Groups. Developers can create and modify applications and browse database objects in an allocated workspace and schema. The retailer workspace schema has privileges required by Oracle Data Mining for executing analytic models.

Figure 21-2 Manage Users and Groups



1. Create User. This area displays the Oracle APEX Create User screen that can be used to create a new Developer account and assign a RESTful Service group. Note that this user must also be created in Identity Management with the same username and password.

Figure 21-3 Create User

Manage Users and Groups \ Create User

Create User

Cancel Create and Create Another Create User

Show All User Identification Account Privileges Password Group Assignments

User Identification

Username RETWSP_DEVELOPER

Email Address RETWSP_DEVELOPER@retailer.com

First Name

Last Name

Description

Default Date Format

Users

Use this page to control access for Application Express application developers, workspace administrators and end users.

Developers can create and edit applications as well as create and modify database objects.

Workspace administrators can additionally create and edit user accounts, manage groups, manage development services.

End users have no development privileges and are defined to provide access control to applications that do not use an external authentication scheme.

2. Assign Group RESTful Service.

Figure 21-4 Assign Group

Manage Users and Groups \ Create User

Create User

Cancel Create and Create Another Create User

Show All User Identification Account Privileges Password Group Assignments

User Identification

Username RETWSP_DEVELOPER

Email Address RETWSP_DEVELOPER@retailer.com

First Name

Last Name

Description

Default Date Format

Users

Use this page to control access for Application Express application developers, workspace administrators and end users.

Developers can create and edit applications as well as create and modify database objects.

Workspace administrators can additionally create and edit user accounts, manage groups, manage development services.

End users have no development privileges and are defined to provide access control to applications that do not use an external authentication scheme.

3. Set Require Change of Password on First Use to No.

Figure 21-5 Require Change of Password

Create User

Cancel Create and Create Another Create User

Show All User Identification Account Privileges Password (For authentication against w... Group Assignments

Require Change of Password on First Use No

end users.

Developers can create and edit applications as well as create and modify database objects.

Workspace administrators can additionally create and edit user accounts, manage groups, manage development services.

End users have no development privileges and are defined to provide access control to applications that do not use an external authentication scheme.

SQL Workshop

The SQL Workshop provides tools to view and manage database objects. To create new database objects, click **Object Browser** and then click **Create**.

Figure 21-6 SQL Workshop Object Browser: Reading Database Objects

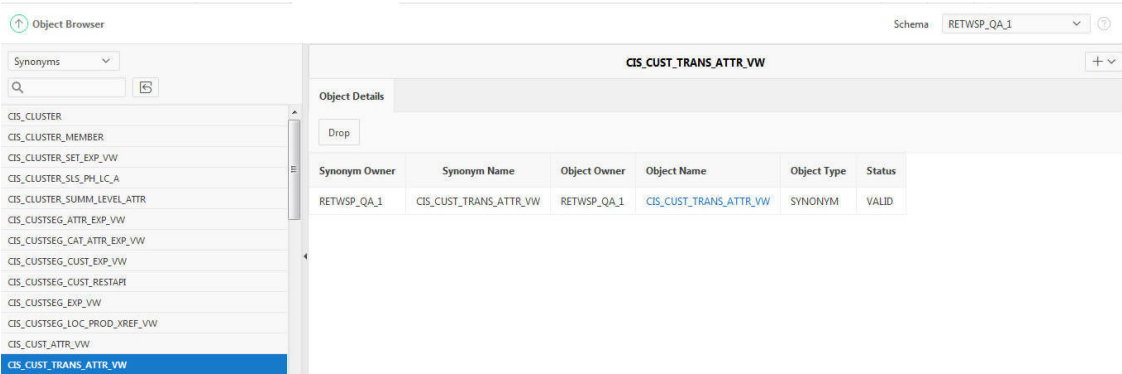


Figure 21-7 SQL Workshop Create Object

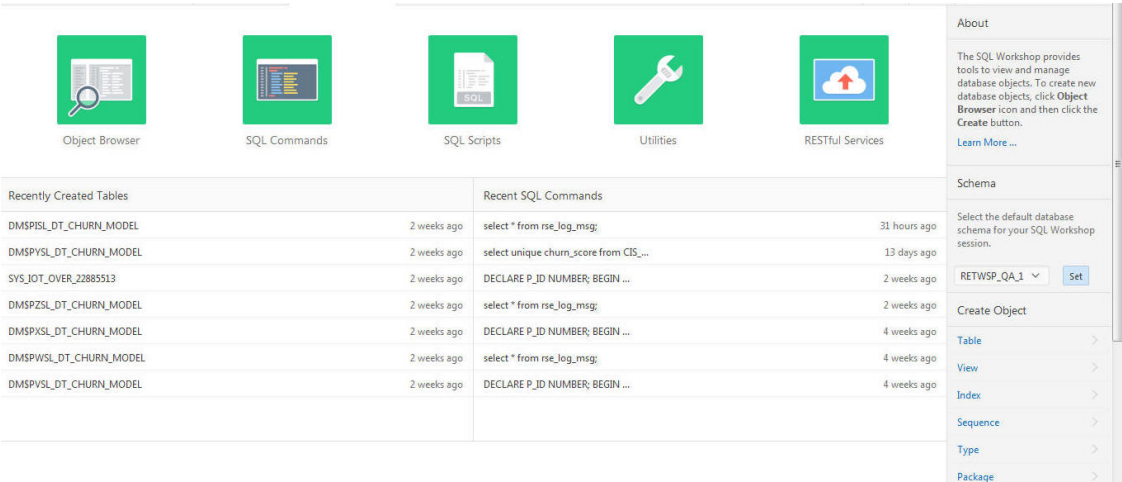


Figure 21-8 SQL Workshop SQL Command: Executing Ad Hoc SQLs

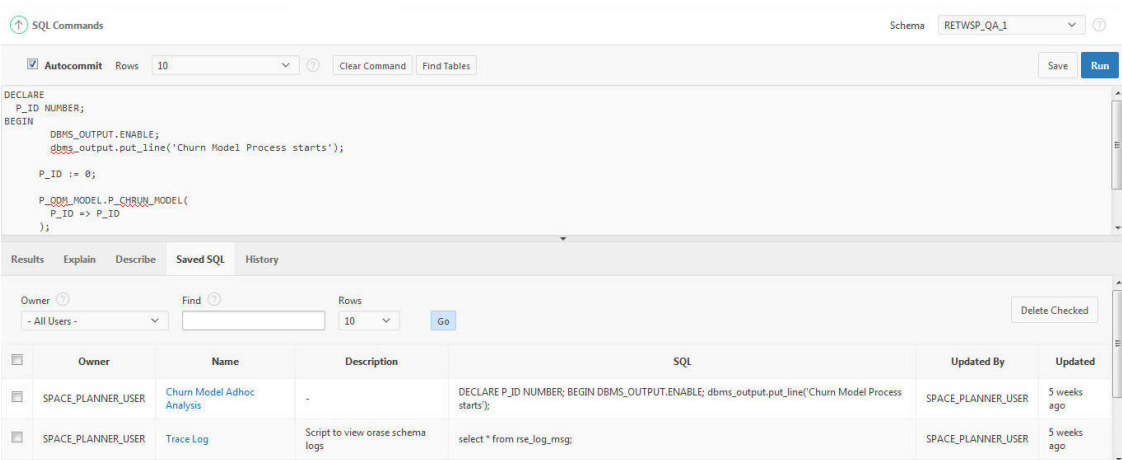
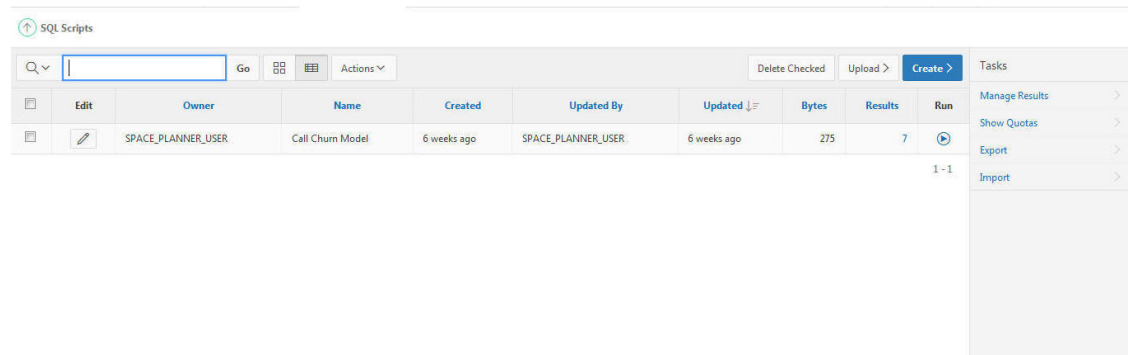
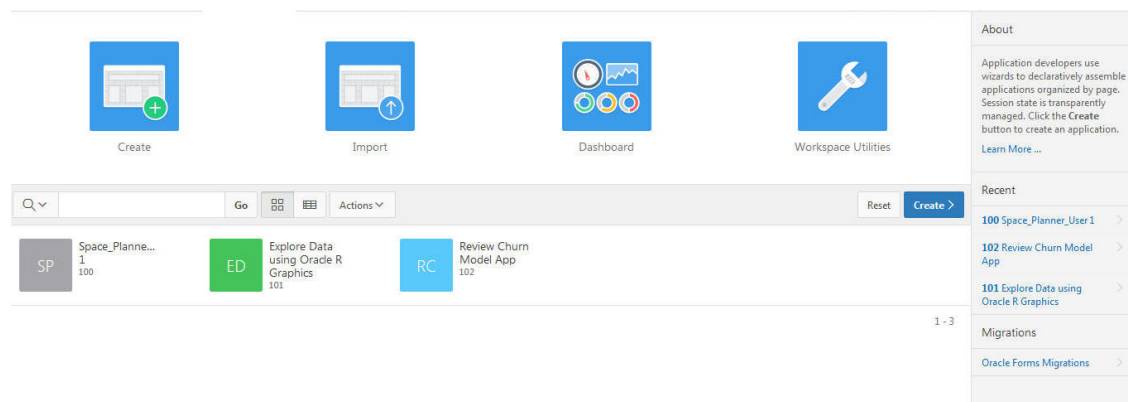


Figure 21-9 SQL Workshop SQL Scripts: Uploading, Executing, and Running

Application Builder

Application developers use wizards to declaratively assemble applications organized by page. The session state is transparently managed. Click **Create** to create an application.

Figure 21-10 Application Builder

Oracle Advanced Analytics

Oracle Advanced Analytics has, as one of the components in Oracle Database Enterprise Edition, Oracle Data Mining. Innovation Workbench assigns privileges required by Oracle Data Mining for execution.

Oracle Data Mining offers a comprehensive set of in-database algorithms for performing a variety of mining tasks, such as classification, regression, anomaly detection, feature extraction, clustering, and affinity analysis.

Figure 21-11 Machine Learning Algorithms in Database

Classification • Decision Tree • Logistic Regression • Naïve Bayes • Support Vector Machine • RandomForest	Clustering • Hierarchical k-Means • Orthogonal Partitioning Clustering	Market Basket Analysis • Apriori – Association Rules
Regression • Linear Model • Generalized Linear Model • Multi-Layer Neural Networks • Stepwise Linear Regression • Support Vector Machine	Attribute Importance • Minimum Description Length	Feature Extraction • Nonnegative Matrix Factorization • Principal Component Analysis • Singular Value Decomposition
	Anomaly Detection • 1 Class Support Vector Machine	Time Series • Single Exponential Smoothing • Double Exponential Smoothing

Oracle Data Mining

Oracle Data Mining can be used to build and deploy predictive and descriptive data mining applications, add intelligent capabilities to existing applications, and generate predictive queries for data exploration.

The Oracle Data Mining developers guide, samples, and tutorials are available at the following websites.

Oracle Data Mining Developer's Guide

http://www.oracle.com/pls/db121/vbook_subject?subject=dma

Data Mining Concepts

<https://docs.oracle.com/database/121/DMCON/toc.htm>

Oracle Data Mining Sample Programs

<http://www.oracle.com/technetwork/database/options/advanced-analytics/odm/odm-samples-194497.html>

Samples can be downloaded odm12csampleprograms-2184025.7z

How to Invoke Oracle Data Mining

The following scripts show how to invoke Oracle Data Mining Classification script that creates a classification model using the Decision Tree algorithm.

```
CREATE OR REPLACE PACKAGE BODY pkg_odm_model
AS
  -- DESCRIPTION - This script creates a classification model using the Decision Tree
  algorithm.
  PROCEDURE proc_churn_model(
    p_id NUMBER)
  IS
  BEGIN
    DECLARE ----- start drop RETWSP_CUST_CHURN_MODEL
      not_found EXCEPTION;
```

```

        PRAGMA EXCEPTION_INIT(not_found, -40203);
BEGIN
    dbms_output.put_line('Start Drop RETWSP_CUST_CHURN_MODEL Tables');
    DBMS_DATA_MINING.DROP_MODEL('RETWSP_CUST_CHURN_MODEL');
    dbms_output.put_line('End Drop RETWSP_CUST_CHURN_MODEL Tables');
EXCEPTION
WHEN not_found THEN
    dbms_output.put_line('RETWSP_CUST_CHURN_MODEL not found');
END; ----- end drop RETWSP_CUST_CHURN_MODEL -----
-- CREATE A SETTINGS TABLE
--
-- The default classification algorithm is Naive Bayes. In order to override
-- this, create and populate a settings table to be used as input for
-- CREATE_MODEL.
--
DECLARE ----- start drop RETWSP_CUST_CHMDL_SETTINGS
    not_found EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_found, -40203);
BEGIN
    dbms_output.put_line('Start Drop RETWSP_CUST_CHMDL_SETTINGS Tables');
    EXECUTE IMMEDIATE 'DROP TABLE RETWSP_CUST_CHMDL_SETTINGS';
    dbms_output.put_line('End Drop RETWSP_CUST_CHMDL_SETTINGS Tables');
EXCEPTION
WHEN not_found THEN
    dbms_output.put_line('RETWSP_CUST_CHMDL_SETTINGS not found');
END; ----- end drop RETWSP_CUST_CHMDL_SETTINGS
DECLARE ----- start drop RETWSP_CUST_CHMDL_COST
    not_found EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_found, -40203);
BEGIN
    dbms_output.put_line('Start Drop RETWSP_CUST_CHMDL_COST Tables');
    EXECUTE IMMEDIATE 'DROP TABLE RETWSP_CUST_CHMDL_COST';
    dbms_output.put_line('End Drop RETWSP_CUST_CHMDL_COST Tables');
EXCEPTION
WHEN not_found THEN
    dbms_output.put_line('RETWSP_CUST_CHMDL_COST not found');
END; ----- end drop RETWSP_CUST_CHMDL_COST
DECLARE ----- start create table RETWSP_CUST_CHMDL_SETTINGS
    already_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(already_exists, -00955);
BEGIN
    dbms_output.put_line('Start Create RETWSP_CUST_CHMDL_SETTINGS Tables');
    EXECUTE IMMEDIATE 'CREATE TABLE RETWSP_CUST_CHMDL_SETTINGS (
setting_name  VARCHAR2(30),
setting_value VARCHAR2(4000))';
    dbms_output.put_line('End Create RETWSP_CUST_CHMDL_SETTINGS Tables');
EXCEPTION
WHEN already_exists THEN
    dbms_output.put_line('Exception not found');
END; ----- end create table RETWSP_CUST_CHMDL_SETTINGS
DECLARE ----- Create RETWSP_CUST_CHMDL_COST Tables begins
    already_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(already_exists, -00955);
BEGIN
    dbms_output.put_line('Start Create RETWSP_CUST_CHMDL_COST Tables');
    EXECUTE IMMEDIATE 'CREATE TABLE RETWSP_CUST_CHMDL_COST (
actual_target_value      NUMBER,
predicted_target_value   NUMBER,
cost                     NUMBER)';
    dbms_output.put_line('End Create RETWSP_CUST_CHMDL_COST Tables');
EXCEPTION
WHEN already_exists THEN

```

```

        dbms_output.put_line('RETWSP_CUST_CHMDL_COST not found');
END; ----- Create RETWSP_CUST_CHMDL_COST Tables ends
-- CREATE AND POPULATE A COST MATRIX TABLE
--
-- A cost matrix is used to influence the weighting of misclassification
-- during model creation (and scoring).
-- See Oracle Data Mining Concepts Guide for more details.
--
dbms_output.put_line('Start Insert records into RETWSP_CUST_CHMDL_COST');
DECLARE ----- sub-block begins
    already_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(already_exists, -00955);
BEGIN
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (0,0,0)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (0,1,1)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (0,2,2)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (0,3,3)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (1,0,1)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (1,1,0)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (1,2,2)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (1,3,3)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (2,0,3)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (2,1,2)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (2,2,0)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (2,3,1)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (3,0,3)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (3,1,2)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (3,2,1)';
    EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_COST VALUES (3,3,0)';
    dbms_output.put_line('End Insert Records');
EXCEPTION
WHEN already_exists THEN
    dbms_output.put_line('RETWSP_CUST_CHMDL_COST not found');
END; ----- sub-block ends
dbms_output.put_line('End Insert records into RETWSP_CUST_CHMDL_COST');
-- Populate settings table
DECLARE ----- sub-block begins
    already_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(already_exists, -00955);
    v_stmt          VARCHAR2(4000);
    v_algo_name      VARCHAR2(100);
    v_algo_decision_tree VARCHAR2(100);
BEGIN
    dbms_output.put_line('Start Populate settings table' || dbms_data_mining.algo_name);
    dbms_output.put_line('Start Populate settings table' ||
dbms_data_mining.algo_decision_tree);
    v_algo_name      := dbms_data_mining.algo_name;
    v_algo_decision_tree := dbms_data_mining.algo_decision_tree;
    v_stmt           := 'INSERT INTO RETWSP_CUST_CHMDL_SETTINGS (setting_name,
setting_value) VALUES ('' || v_algo_name || ''',' || v_algo_decision_tree || '')';
    dbms_output.put_line('Start Populate settings table v_stmt --' || v_stmt);
    EXECUTE IMMEDIATE v_stmt;
EXCEPTION
WHEN already_exists THEN
    dbms_output.put_line('Exception not found');
END; ----- sub-block ends
DECLARE ----- sub-block begins
    already_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(already_exists, -00955);
    v_table_name     VARCHAR2(100);
    v_matrix_cost     VARCHAR2(100);
BEGIN

```



```

v_table_name := dbms_data_mining.clas_cost_table_name;
v_matrix_cost := 'RETWSP_CUST_CHMDL_COST';
EXECUTE IMMEDIATE 'INSERT INTO RETWSP_CUST_CHMDL_SETTINGS (setting_name,
setting_value) VALUES' || '(' || v_table_name || ',' || v_matrix_cost || ')';
dbms_output.put_line('End Populate settings table');
EXCEPTION
WHEN already_exists THEN
dbms_output.put_line('Exception not found');
END; ----- sub-block ends
-----
-- CREATE A NEW MODEL
--
-- Build a DT model
dbms_output.put_line('Start Create Churn Model');
DBMS_DATA_MINING.CREATE_MODEL( model_name => 'RETWSP_CUST_CHURN_MODEL',
mining_function => dbms_data_mining.classification, data_table_name =>
'cis_cust_attr_vw', case_id_column_name => 'CUSTOMER_ID', target_column_name =>
'CHURN_SCORE', settings_table_name => 'RETWSP_CUST_CHMDL_SETTINGS');
dbms_output.put_line('End Create Churn Model');
-----
-- DISPLAY MODEL SIGNATURE
--
column attribute_name format a40
column attribute_type format a20
SELECT attribute_name,
attribute_type
FROM user_mining_model_attributes
WHERE model_name = 'RETWSP_CUST_CHURN_MODEL'
ORDER BY attribute_name;
END p_chrun_model;
END pkg_odm_model;

```

Test ODM Model

```

DECLARE
RUN_ID NUMBER;
BEGIN
DBMS_OUTPUT.ENABLE;
dbms_output.put_line('Churn Model Process starts');
RUN_ID := 1001;
pkg_odm_model.proc_churn_model( RUN_ID => RUN_ID );
dbms_output.put_line('Churn Model Process ends');
END;

```

Notebooks

Data scientists can use the Innovation Workbench Notebook to create notebooks, which are collections of documentation, snippets of code, and visualizations. These notebooks are bundled with key python modules for machine learning, data mining, natural language processing, network analysis, and optimization solvers.

Here is a list of some of the python packages that are bundled with AIF. These packages provide features that span data mining processes of from data exploration to data visualization.

- Ensemble Machine Learning Algorithms with scikit-learn
- Data exploration and analysis using Pandas; NumPy; SciPy
- Data visualization using Matplotlib; Seaborn
- Data storage using cx_Oracle

- Graph algorithms using Networkx
- Optimization using Gurobi Solver

The full list of python libraries bundled with AIF can be viewed from a notebook by running the following:

```
print("=====")
print("Installed Libraries:")
print("=====")
from pip._internal.operations.freeze import freeze
for requirement in freeze(local_only=True):
    print(requirement)
quit ()
pip list
```

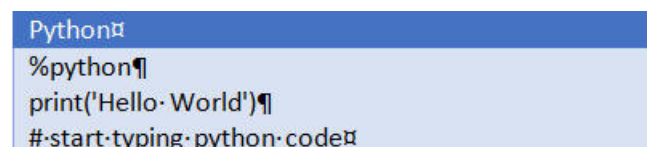
Note that, due to security requirements, in some instances previously packaged libraries have been deprecated.

Data Studio graph analytics includes numerous built-in graph algorithms. Some of the classes of algorithms that it provides include:

- Community Detection
- Path Finding
- Ranking
- Recommendation
- Pattern Matching
- Influencer Identification

Invoking Python Code

Figure 21-12 Invoking Python Code

A screenshot of a Jupyter Notebook cell. The cell has a blue header bar with the text "Python". Below the header, the code is as follows:

```
%python
print('Hello World')
#start typing python code
```

Connecting to a database using cx_Oracle

Database connection string should be fetched from environment variable. Refer to code in [Figure 21-13](#) in red.

Figure 21-13 Connecting to a Database Using cx_Oracle

```
Python#
Cx_oracle#
%python#
import cx_Oracle#
import os#
import pandas as pd#
#
dbwallet_entry=os.environ['PYTHON_RETWSP_DBALIAS']#
print(dbwallet_entry)#
con=cx_Oracle.connect('/'+dbwallet_entry)#
ver=con.version.split(".")#
print('Version')#
print(ver)#
#
query="SELECT * FROM rse_prod_hier where rownum < 5"#
df_ora=pd.read_sql(query, con=con)#
df_ora.iloc[0]#
#
```

Executing Gurobi code

Create a Gurobi environment by calling `RseGurobiEnv.getGurobiEnv()`. Refer to the code in red in [Figure 21-14](#)

Figure 21-14 Executing Gurobi Code

```

Python
Gurobi
%python
from gurobipy import *
from rsecommon.analytics.gurobi import *
try:
    instance = rseenv.RseGurobiEnv.getInstance()
    env = instance.getGurobiEnv()
    m = Model("mip1", env)

    # Create variables
    x = m.addVar(vtype=GRB.BINARY, name="x")
    y = m.addVar(vtype=GRB.BINARY, name="y")
    z = m.addVar(vtype=GRB.BINARY, name="z")

    # Set objective
    m.setObjective(x + y + 2 * z, GRB.MAXIMIZE)

    # Add constraint: x + 2 * y + 3 * z <= 4
    m.addConstr(x + 2 * y + 3 * z <= 4, "c0")

    # Add constraint: x + y >= 1
    m.addConstr(x + y >= 1, "c1")

    m.optimize()

    for v in m.getVars():
        print('%s %g' % (v.varName, v.x))

    print('Obj: %g' % m.objVal)

except GurobiError as e:
    print('Error code: ' + str(e.errno) + ": " + str(e.message))

except AttributeError:
    print('Encountered an attribute error')

```

Invoking Oracle SQL/PL/SQL procedures

Figure 21-15 Invoking Oracle SQL/PL/SQL Procedures

```

OracleJDBC
%oracle

```

Executing SQL

Figure 21-16 Executing SQL

```
Execute-SQL␣  
%oracle␣  
select * from RSE_LOC_HIER;␣
```

Executing PL/SQL

Figure 21-17 Executing PL/SQL

```
Execute-PLSQL␣  
%oracle␣  
call DBMS_OUTPUT.PUT_LINE('I am plsql procedure, use call procedure_name');␣
```

Invoking Parallel Graph Analytics

Figure 21-18 Invoking Parallel Graph Analytics

```
Parallel-Graph-Analytics␣  
%pgx␣
```

Creating a graph from scratch

Figure 21-19 Graph Creation

```

Build-a-graph-from-scratch
%pgx
builder=session.newGraphBuilder()
//create-a-few-vertices
v1=builder.addVertex(1).addLabel("Person").setProperty("name","Charles").setProperty("age",24)
v2=builder.addVertex(2).addLabel("Person").setProperty("name","Susan").setProperty("age",36)
v3=builder.addVertex(3).addLabel("Person").setProperty("name","Philip").setProperty("age",22)
v4=builder.addVertex(4).addLabel("Place").setProperty("name","McDonalds")
v5=builder.addVertex(5).addLabel("Place").setProperty("name","Wendy's")
v6=builder.addVertex(6).addLabel("Place").setProperty("name","Office")
v7=builder.addVertex(7).addLabel("Car").setProperty("name","VW-Passat")
v8=builder.addVertex(8).addLabel("Car").setProperty("name","Toyota-Highlander")
v9=builder.addVertex(9).addLabel("Place").setProperty("name","Central-Park")
//vertices-can-have-multiple-labels
v4.addLabel("Restaurant")
v5.addLabel("Restaurant")
//create-a-few-edges
builder.addEdge(0,v1,v2).setLabel("knows")
builder.addEdge(1,v2,v1).setLabel("knows")
builder.addEdge(2,v2,v3).setLabel("knows")
builder.addEdge(3,v4,v5).setLabel("connected")
builder.addEdge(4,v5,v6).setLabel("connected")
builder.addEdge(5,v4,v6).setLabel("connected")
builder.addEdge(6,v1,v7).setLabel("owns").setProperty("since",1998)
builder.addEdge(7,v2,v8).setLabel("owns").setProperty("since",2003)
builder.addEdge(8,v1,v6).setLabel("worksAt")
builder.addEdge(9,v2,v6).setLabel("worksAt")
builder.addEdge(10,v9,v4).setLabel("connected")
//store-resulting-graph-in-a-variable-'graph'
graph=builder.build()

```

Scheduling Innovation Workbench Python Notebook

This section describes the scheduling process.

IDCS Setup

In order to use the Datastudio Notebook REST API, the Oracle Identity Cloud Service (IDCS) application is necessary. The IDCS application must be created with following grant_type as (password) and client_credentials.

Once the IDCS application is created, a new clientId:clientSecret will be generated. Note these credentials in order to perform a REST API request. A user must also be created in order to set up the Datastudio User Session.

Scheduling Jobs

Notebooks can be scheduled for automatic execution. To implement this, make a POST request to the using REST API call with the following payloads.

Execute once, immediately.

```
{
  "cronSchedule": "",
  "timeEnd": "",
}
```

Execute at a regular interval, and stop at a given date.

```
{
  "cronSchedule": "0/1440 * * * * *",
  "timeEnd": "2021-02-24T21:10:34.400Z",
  "id": "dsYVGG9Mvw"
}
```

Execute at a regular interval, indefinitely.

```
{
  "cronSchedule": "0/ 1440 * * * * *",
  "timeEnd": "",
  "id": "dsYVGG9Mvw"
}
```

REST API Documentation

The REST API documentation details for request/response are located here:<http://datastudio.oraclecorp.com/docs/apidoc/swagger-ui.html#/>

Example

Here is a REST API curl call example showing how to schedule a notebook.

Table 21-1 Example for Scheduling Notebooks

Seq. #	REST API Purpose	REST API Curl Request	REST API Curl Response
1	Fetch Authentication Token POST request	curl --location --request POST 'https://<IDCS_HOST>/oauth2/v1/token' \--header 'Authorization: Basic <Base64 ClientID:ClientSecret>' \--header 'Content-Type: application/x-www-form-urlencoded' \--data-urlencode 'grant_type=password' \--data-urlencode 'scope=urn:opc:ldm:__myscopes__' \--data-urlencode 'username=<username>' \--data-urlencode 'password=<password>'	Response will have authentication token. To make subsequent request: { "access_token": "<Access Token>", "token_type": "Bearer", "expires_in": 3600}

Table 21-1 (Cont.) Example for Scheduling Notebooks

Seq. #	REST API Purpose	REST API Curl Request	REST API Curl Response
2	Setup SessionGET request	curl --location --request GET ' <u><APP_HOST>/datastudio/v2/sessions/user</u> ' --header 'Authorization: Bearer <u><Token></u> '	{ "username": " <u><user_name></u> ", "permissions": ["graph_create", "export_all", "view_permissions_tab", "import_notebook", "view_dashboard_tab", "view_credentials_tab", "create_notebook", "create_credential", "view_interpreter_tab", "delete_all"], "authToken": " <u><Token></u> " }
Note: The first two steps above are mandatory. Execute any of the following steps.			
3	Schedule NotebookPOST request	curl --location --request POST ' <u>https://<APP_HOST>/datastudio/v2/notebooks/schedule</u> ' --header 'Authorization: Bearer <u><Token></u> ' --header 'Content-Type: application/json' --data-raw '{ "cronSchedule": "", "timeEnd": "", "id": " <u>dsYVGG9Mvw</u> " }'	{ "id": " <u>dsja4YWg</u> ", "status": "SUBMITTED", "startTime": null, "endTime": null, "error": null, "tasks": [] }
4	Schedule Notebook Paragraphs to execute immediatelyPOST request	curl --location --request POST ' <u>https://<APP_HOST>/datastudio/v2/notebooks/schedule</u> ' --header 'Authorization: Bearer <u><Token></u> ' --header 'Content-Type: application/json' --data-raw '{ "cronSchedule": "", "paragraphs": [{ "id": " <u>dsB0zM58</u> " }], "timeEnd": "", "id": " <u>dsYVGG9Mvw</u> " }'	{ "id": " <u>dsVBqIO3</u> ", "status": "SUBMITTED", "startTime": null, "endTime": null, "error": null, "tasks": [] }
5	Schedule Notebook to execute immediately.Execute a notebook every five minutes.POST request	curl --location --request POST ' <u>https://<APP_HOST>/datastudio/v2/notebooks/schedule</u> ' --header 'Authorization: Bearer <u><Token></u> ' --header 'Content-Type: application/json' --data-raw '{ "cronSchedule": " <u>*/1440 * * * * *</u> ", "endDate": " <u>2020-12-17T14:10:34.400Z</u> ", "id": " <u>dsYVGG9Mvw</u> " }'	{ "id": " <u>dsja4YWg</u> ", "status": "SUBMITTED", "startTime": null, "endTime": null, "error": null, "tasks": [] }
6	Schedule Notebook Paragraphs.Execute a notebook/paragraph every five minutes.POST request	curl --location --request POST ' <u>https://<APP_HOST>/datastudio/v2/notebooks/schedule</u> ' --header 'Authorization: Bearer <u><Token></u> ' --header 'Content-Type: application/json' --data-raw '{ "cronSchedule": " <u>*/1440 * * * * *</u> ", "paragraphs": [{ "id": " <u>dsB0zM58</u> " }], "timeEnd": " <u>2020-12-17T14:10:34.400Z</u> ", "id": " <u>dsYVGG9Mvw</u> " }'	{ "id": " <u>dsVBqIO3</u> ", "status": "SUBMITTED", "startTime": null, "endTime": null, "error": null, "tasks": [] }

Table 21-1 (Cont.) Example for Scheduling Notebooks

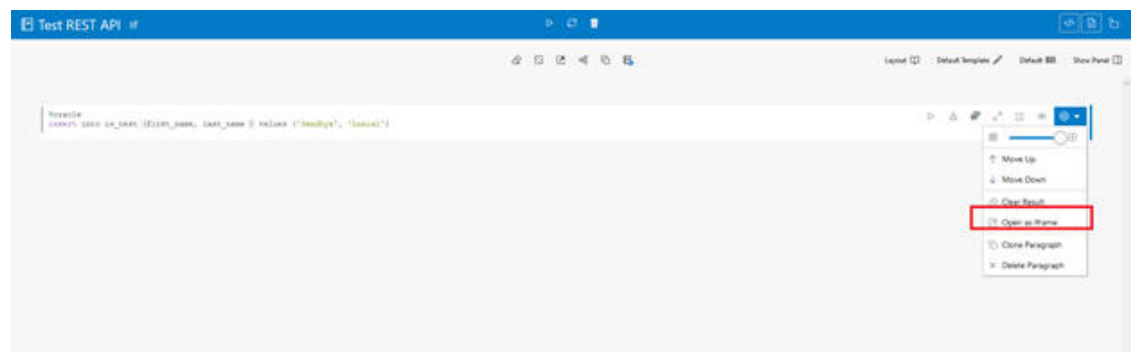
Seq. #	REST API Purpose	REST API Curl Request	REST API Curl Response
7	Schedule Notebook Paragraphs. Execute a notebook/paragraph adhoc with parameters.	curl --location --request POST 'https://<hostname>/datastudio/v2/notebooks/schedule' \--header 'Authorization: Bearer <Token>' \--header 'Content-Type: application/json' \--data-raw '{ "cronSchedule": "", "paragraphs": [{ "id": "dsa7kZYv", "message": "string", "params": { "FirstName": "Sandhya", "LastName": "Lonial" } }], "timeEnd": "2020-02-24T21:10:34.400Z", "id": "dsYVGG9Mvw" }	{ "id": "dsOB0yra", "status": "SUBMITTED", "startTime": null, "endTime": null, "error": null, "tasks": [] }
8	Schedule Notebook Paragraphs. Execute a multiple paragraphs.	{ "cronSchedule": "", "paragraphs": [{ "id": "dsB0zM58" }, { "id": "dsa7kZYv", "message": "string", "params": { "FirstName": "Sandhya", "LastName": "Lonial" } }], "timeEnd": "2020-02-24T21:10:34.400Z", "id": "dsYVGG9Mvw" }	{ "id": "dsOB0yra", "status": "SUBMITTED", "startTime": null, "endTime": null, "error": null, "tasks": [] }

Notebook and Paragraph IDs

To obtain notebook and paragraph IDs, select a paragraph, using the Setting Context menu. Click **Open as IFrame**. A new browser tab opens. Copy the url in the address bar in browser and borrow IDs from the URL notebookId=dsYVGG9Mvw&pid=dsB0zM58

https://<host>/datastudio/?
root=iParagraph&notebookId=dsYVGG9Mvw&pid=dsB0zM58&readOnly=false&showCode=true

Figure 21-20 Notebook and Paragraph IDs



Considerations

Consider the following:

- Make sure there are no empty paragraphs as this results in an 500 error response.

- To test, create a table and view records that are inserted in the table.

```
%oracle
insert into iw_test (first_name, last_name ) values (John, 'Doe')

%python
print('Hello World')
```

Verify using the following:

```
CREATE TABLE iw_test (
  person_id NUMBER GENERATED BY DEFAULT AS IDENTITY,
  first_name VARCHAR2(50) NOT NULL,
  last_name VARCHAR2(50) NOT NULL,
  PRIMARY KEY(person_id)
);
select count(*) from iw_test;
```

REST API

Here is a REST API curl call example showing how to schedule a notebook.

Table 21-2 Example

REST API purpose	REST API curl request	REST API curl response
Fetch Authentication Token	curl --location --request POST 'http://bur00afq.us.oracle.com:26000/datastudio/v2/sessions/login' \--header 'Authorization: Basic b3Jhc2UxOnBhc3N3b3JkMQ==' \--header 'Content-Type: application/json' \--header 'Cookie: JSESSIONID=cmWRE8eaMH1e7CIA72Qx8QVxX3SKVMuEi1S_ErDgySh2qInJRbdZ!1637160649' \--data-raw '{"credentials":"password1","principal":"orase1"}'	Response will have authentication token. To make subsequent request:authToken":"de6460f5-e36c-4592-b2c3-6ef18268e6c5"
Schedule Notebook	curl -X POST -H 'Content-Type: application/json' -H 'x-auth-token: de6460f5-e36c-4592-b2c3-6ef18268e6c5' -i 'https://<host>:<port>/datastudio/v2/notebooks/schedule' --data '{ "cronSchedule": "0/5 * * * *", "endDate": "2020-02-24T21:10:34.400Z", "id": "dsZgvK3G"}	{"id":"dsbBVLgL","status":"SUBMITTED","startTime":null,"endTime":null,"error":null,"tasks":[]}

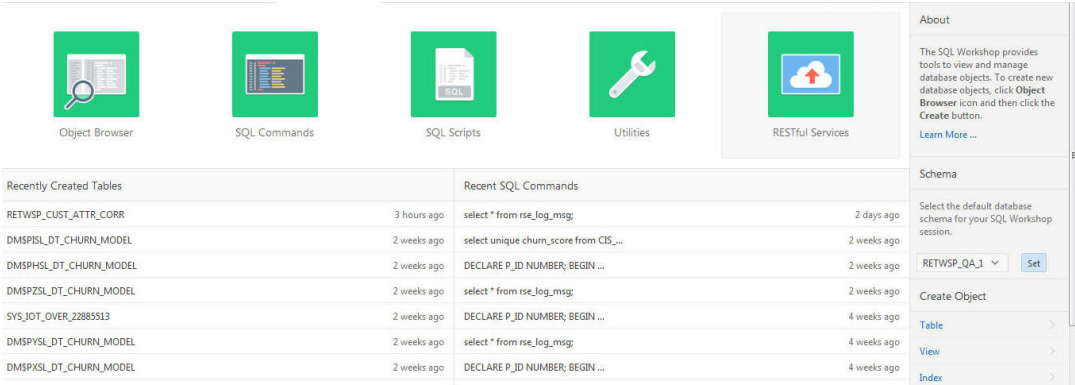
Restful Service

RESTful Services allow for the declarative specification of RESTful access to the database. They are created by configuring a set of Uniform Resource Identifiers (URIs) to a SQL query or anonymous PL/SQL block. The set of URIs is identified by a URI template.

To create a RESTful service:

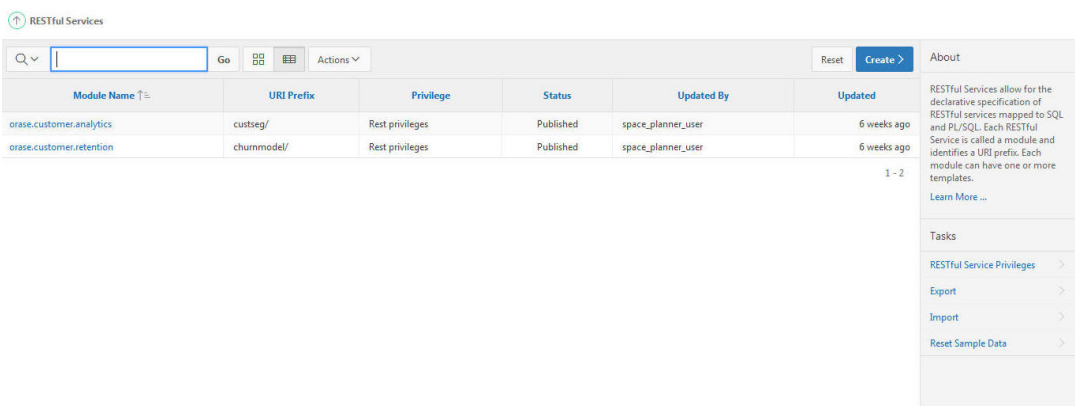
1. Select SQL Workshop RESTful Services.

Figure 21-21 RESTful Service



2. On selecting RESTful Services, you see the option to create a RESTful Service.

Figure 21-22 Create RESTful Service



3. A RESTful Service Module is a grouping of common templates, known as Resource Templates, under a common URI prefix. This prefix is prepended to all templates.

Figure 21-23 Create Module



4. A URI template is a simple syntax for describing URIs. You populate the required fields as shown in [Figure 21-24](#) and make sure to set the required privileges as Rest Privileges.

Figure 21-24 Resource Template

Resource Template

Add a Resource Template

URI Template:

Priority:

Entity Tag:

or an anonymous PL/SQL block responsible for handling a particular HTTP method. Multiple handlers can be defined for a Resource Template, however only one handler per HTTP method is permitted.

5. A Resource Handler is a query or an anonymous PL/SQL block responsible for handling a particular HTTP method. Multiple handlers can be defined for a Resource Template; however, only one handler per HTTP method is permitted. You can select method, source type, format, and SQL Query to read data from the schema.

Figure 21-25 Resource Handler

Resource Handler

Add a Resource Handler

Method:

Source Type:

Format:

Source:

6. To test the rest API, click **Test Button** (see Test at the bottom right corner.)

Figure 21-26 Test the REST API

RESTful Services - RESTful Service Module - Resource Template - Resource Handler

Resource Handler: GET

RESTful Service Module:

URI Template:

Method:

Source Type:

Format:

Requires Secure Access:

Pagination Size:

Source

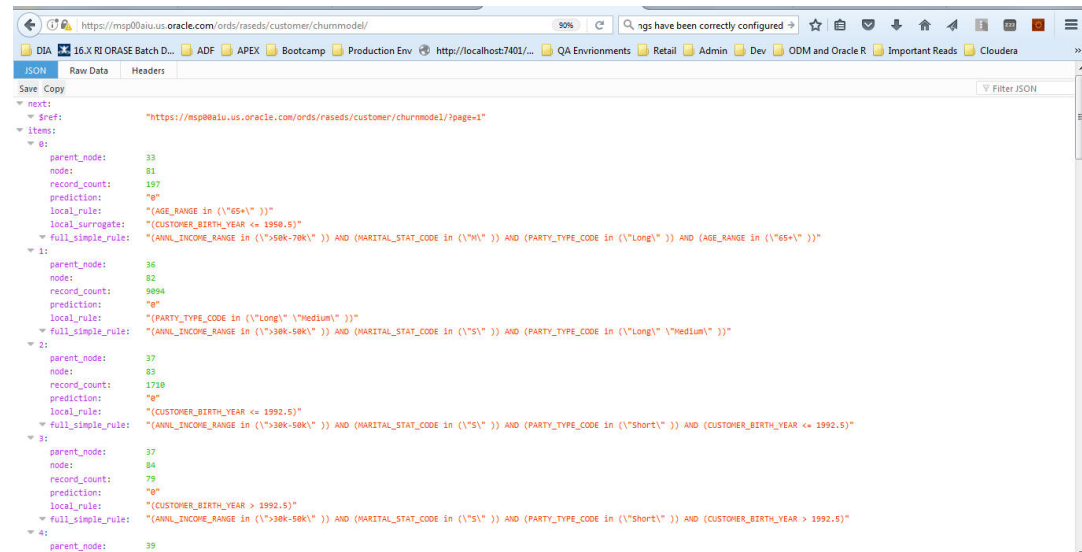
Source:

Example

Test

7. The page refreshes and displays data in JSON format.

Figure 21-27 RESTful Refresh



Integration

IW provides a customer with a way to integrate with other applications. A framework is provided that enables it to receive and read-in files from customers so that the data contained can be loaded into IW tables. They are thus able to use data from their other systems within IW itself.

Consequently, data within IW can also be exported to flat files so that their other applications can use the AIF-cleansed data that is available via IW.

More details on the framework for these capabilities are contained in the following Custom Data Loads and Custom Data Exports sections.

Custom Data Loads

AIF provides functionality to load data files from a customer site sent via Secure FTP, from which all other daily interface files are sent. A zip file, which can contain any number of files, can be provided, along with the specifications about the files to be loaded. (These files are referred to as context files here.) Once the data has been sent, a process is invoked from the Process Orchestration and Monitoring (POM) application to initiate loading the data into the appropriate tables inside the Innovation Workbench. Once the data load is complete, a zip will be uploaded to Object Storage to provide the data load logs, which will indicate whether or not there were any rejected records.

Required Files

The following are the prerequisites for performing a data load:

- The table must exist inside the Innovation Workbench.
- A zip file named `ORASE_IW_DATA.zip` must be created. It must contain the data to be loaded and the context files that describe details about the data load.
- The `ORASE_IW_DATA.zip` file must be uploaded to the Object Storage using the File Transfer Service (FTS).

Zip File Contents

The ZIP file must contain a pair of files for each table to be loaded, a .dat file and a .ctx file. The data file can have any name, with the .dat extension. The context file must match the name of the data file, but with the .ctx extension. If a data file is provided without a .ctx file, then the data file will be ignored.

For example, to load a table called STG_WKLY_SLS, you require a file called STG_WKLY_SLS.dat to hold the data to be loaded and a file called STG_WKLY_SLS.dat.ctx to describe the details regarding the load.

No support exists for directory names inside the zip file, so if they are provided they will be ignored. For example, you cannot send directory1/STG_WKLY_SLS.dat and directory2/STG_WKLY_SLS.dat because they will end up overriding each other when the zip is extracted.

Although you can provide multiple files to be loaded in a single zip, none of them can be for the same table as another load. The load truncates the data in the table to be loaded, so if the same table is targeted by multiple files, then only the latest file will be loaded. Because this data is truncated prior to each load, you should follow a pattern in which these tables are the staging tables to be used to then populate data in other tables. In this way, the data is verified, which allow for the merging of data, via updates or inserts. This is not possible if you loaded directly to your final table.

Context File Details

The .ctx file provide options for including details that describe the data load. These details are similar to the options used by SQL*Loader, so refer to documentation about the SQL*Loader for any additional details about concepts noted here.

Here is an example of a context file.

```
#TABLE#SLS_WKLY_SLS#
#DELIMITER#|#
#COLUMN#WEEK_DATE#DATE(10) "YYYY-MM-DD"#
#COLUMN#PRODUCT_KEY#
#COLUMN#LOCATION_KEY_FILLER#BOUND FILLER#
#COLUMN#SLS_AMT#
#COLUMN#LOCATION_ID#"some_db_package.lookup_location_id(:LOCATION_KEY_FILLER)"#
#COLUMN#LOAD_DATE#"sysdate"#
```

In the above example contextual file:

- The #TABLE## line is required in order to specify the name of the table to be loaded.
- The #DELIMITER## line is optional, and if not provided, then a | will be considered the default value.
- The table to be loaded will be loaded via a Direct Path load (for efficiency sake). It will truncate the table prior to loading the table. This means that you cannot provide multiple files in a single transmission for the same table.
- The columns can be wrapped in " " if necessary.
- The data in the file must be provided as CHARACTERSET AL32UTF8.
- For the column specifications, the columns must be listed in the order in which they appear in the file. The format is #COLUMN#THE_COLUMN_NAME#Any special load instructions#

- Note that for Date columns, you must provide a proper date specification with the format that you are providing the data in. The example above for WEEK_DATE illustrates how to specify the date in a YYYY-MM-DD format.
- For numeric columns, you do not normally require any additional load instructions.
- If you are providing data that must be used as a BOUND FILLER during the load of another column, then you must specify BOUNDFILLER, just like you would in SQL*Loader.
- The LOCATION_ID example above illustrates how you can refer to a FILLER column, which is then used as a parameter to a function, which then returns the actual value to load to the column.
- If you want an value such as the system date/time to be loaded, then you can specify "sysdate" as shown above for LOAD_DATE column.
- When you are populating columns that are loaded via a special expression (such as LOCATION_ID and LOAD_DATE above), be sure to provide them last in the context file.
- When loading data character data, it maybe be necessary to specify the column like this: CHAR(50). This is commonly required by SQL*Loader when advanced character sets are used.
- The examples shown above for LOCATION_KEY_FILLER and LOCATION_ID are not usual/simple use cases, but are supported. For a better understanding of how that works, refer to the documentation for SQL*Loader.

Data Load Feedback

When the data load is complete, a zip file named ORASE_IW_DATA_extract.zip will be uploaded to Object Storage and it will contain the log and bad files for the data load.

Invoking the Data Load

In order to invoke the load process, you must use the POM application to invoke the following adhoc process:

RSE_LOAD_IW_FILES_ADHOC.

This controls the execution of all these steps. Once the load has been completed, you should be able to use the data as necessary inside the Innovation Workbench. See "[Process Orchestration and Monitoring](#)" for additional details.

Custom Data Exports

You can export data that has been created within the Innovation Workbench workspace. You first configure the table to be exported. A process is then executed that exports the table data to a text file. The data is then gathered into a zip file, and the zip file is moved to the Object Storage for retrieval using File Transfer Service (FTS).

Required Configuration

Using the Data Management menu option (described in *Oracle Retail Science Cloud Services User Guide*), you can manage the tables to be exported, using the formatting shown in [Table 21-3](#). Once in the screen, select the table RSE_CUSTOM_EXP_CFG. Then, add rows and make adjustments to existing rows as required in order to define the tables to be exported.

Table 21-3 Export Table Formatting

Column Name	Data Type	Example	Comments
TABLE_NAME	Character(30)	IW_RESULTS_TABLE	The name of the table to be exported.
FILE_NAME	Character(80)	iw_results_table.csv	The name of the file to be created, excluding any directory names.
DESCR	Character(255)	Results table of XYZ Calculation	Any description to describe the table.
COL_DELIMITER	Character(1)	, (comma)	The character to use as a delimiter between columns.
COL_HEADING_FLG	Character(1)	Y	A Y/N value to indicate if the export should include a heading row (Y) that contains the names of the columns, or not (N).
DATE_FORMAT	Character(30)	yyyy-MM-dd HH:mm:ss	The format to use for any date columns. This format must be a valid format for Java date handling.
ENABLE_FLG	Character(1)	Y	A flag to indicate if this table should be exported (Y) or not (N) This flag can be used to temporarily disable an export, without the need to remove it completely.

Invoking the Export

In order to begin the export process, you must use the POM application to invoke the following AdHoc process:

RSE_IW_EXPORT_FILES_ADHOC / job: RSE_IW_EXPORT_FILES_ADHOC_JOB.

This process controls the execution of the steps to export the data. Once the export has been completed, a zip file named ORASE_IW_EXPORT_extract.zip is created. All files will be named according to the names specified in the RSE_CUSTOM_EXP_CFG table. See ["Process Orchestration and Monitoring"](#) for additional details on how to execute a Standalone/AdHoc Sample Extensibility Use Case process.

Sample Extensibility Use Case

Clients can use IW's framework to address use-cases unique to their business requirements.

Here is one possible example of IW's extensibility.

A customer wants to expose AIF-cleansed sales transaction data from AI Foundation to a third-party system, using an integration software for connecting applications, data, and devices. This enables their other applications to access the exact calculations/AIF-cleansed data that they require.

IW can be extended as outlined below in order to address this use-case:

- Load sales data into AI Foundation using the existing nightly sales data load routine.

- Create a custom PL/SQL package to transform the data as required and load the transformed data into custom tables in IW. For example, sales can be aggregated per customer/transaction. This aggregated data is loaded into the custom IW table, with specific custom keys required to identify the records in external systems.

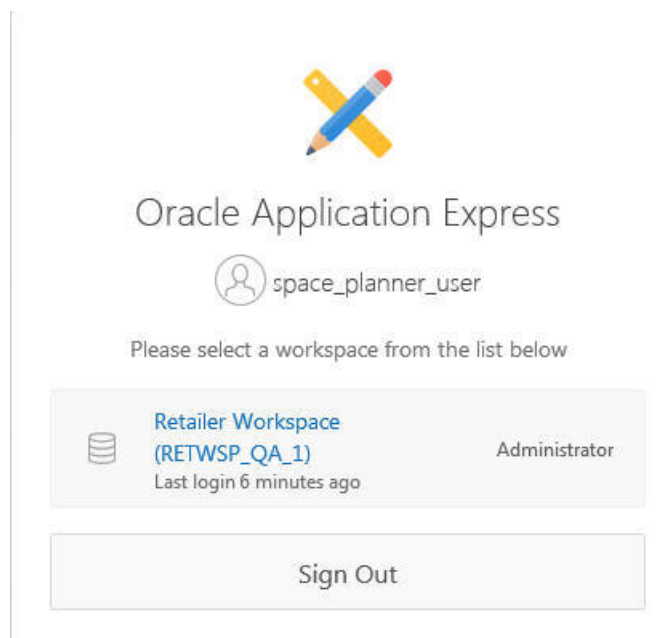
This custom PL/SQL can include status tracking using auxiliary tables to capture the status of the executed routine, as well as log pertinent information that can enable re-execution if necessary.
- Create ReST APIs in APEX that reference the custom IW tables, to expose the sales transaction data.
- Create and schedule load jobs in APEX that invoke the custom PL/SQL package to populate the custom tables. These daily programs must be scheduled using DBMS_SCHEDULER jobs that are aligned with the expected end-time of the daily AIF DATA batch.
- Real-time ReST API calls can then be made from the integration software and then into the third-party system that will use the data.

Troubleshooting

Science Innovation Workbench can also be used for monitoring application logs to troubleshoot issues in an application process due to an error in a batch or business process.

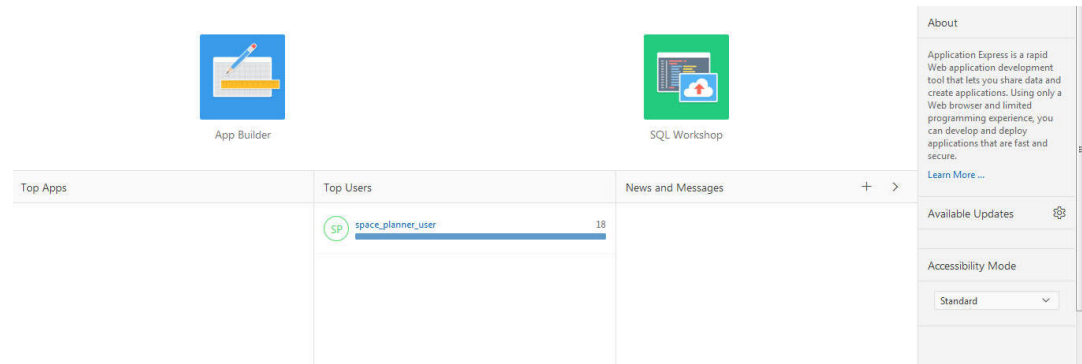
1. Click **Innovation Workbench**.
2. Select **Retailer Workspace**.

Figure 21-28 Retailer Workspace



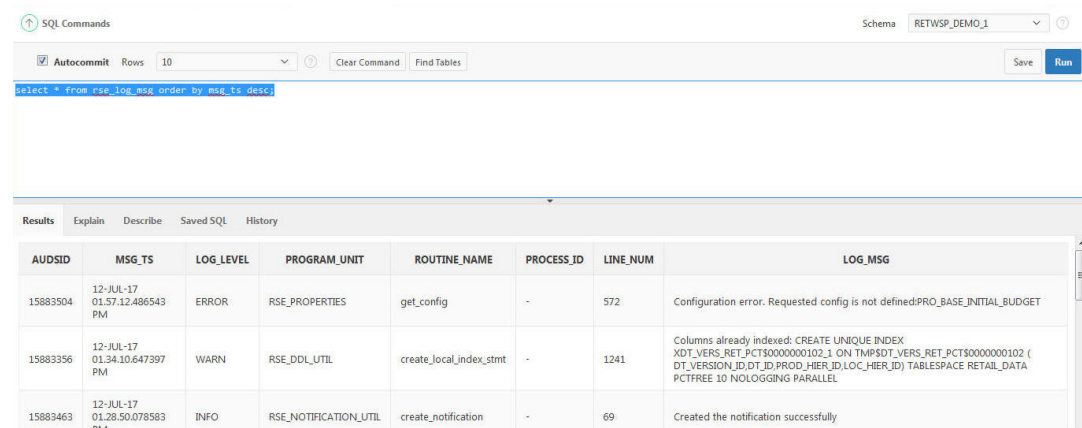
3. Select **SQL Workshop**.

Figure 21-29 SQL Workspace



4. Select **SQL Command**.
 - a. Enter SQL `select * from rse_log_msg order by msg_ts desc;`
 - b. Select run and view the log message in the bottom panel.
 - c. You can describe objects using `describe rse_log_msg;`

Figure 21-30 SQL Command



DBMS Scheduler

Innovation Workbench can be used for data mining tasks that are long running and have to rely on database scheduling functions and procedures that can be called from any PL/SQL program.

Any jobs created in the Retailer Workspace using Innovation Workbench must be created under Job Class `RETAILER_WORKSPACE_JOBS`. Jobs created in a default job class or any other job class other than `RETAILER_WORKSPACE_JOBS` can be disabled by an Oracle Administrator while managing resources.

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
  job_name          => 'retwsp_churn_model',
  job_type          => 'STORED_PROCEDURE',
  job_action        => 'retwsp.pkg_customer_analytics.proc_churn_model',
  start_date        => '01-JAN-17 07.00.00 PM US/Pacific',
  repeat_interval    => 'FREQ=YEARLY; BYDATE=0331,0630,0930,1231; ',
  end_date          => '31-DEC-17 07.00.00 PM US/Pacific',
```

```

        job_class      => 'RETAILER_WORKSPACE_JOBS',
        comments      => 'Retailer workspace churn model job');
END;
/

```

Schema Objects

Database objects owned by the various schemas in AIF applications are available for the advanced analyst to use. Here are some examples:

Table 21-4 Schema Objects

Table Name	Description
rse_cal_hier	This table is used to hold all calendar hierarchies. Examples are the normal calendar hierarchy, and can also contain an alternate hierarchy for the fiscal calendar.
rse_prod_hier	This table is used to hold all product hierarchies. Examples are the normal product hierarchy, and can also contain an alternate category hierarchy.
rse_loc_hier	This table is used to hold all location hierarchies. Examples are the normal organizational hierarchy, and can also contain an alternate hierarchy for trade areas.
rse_prod_loc_status	Status of the item at this location for this time frame. A-Active; I-Inactive; C-Discontinued; D-Deleted
rse_ret_lc_wk_a	This table contains aggregate sales data for the dimensions of a location and a week.
rse_sls_lc_wk_a	This table contains aggregate sales data for the dimensions of a location and a week.
rse_sls_pr_lc_cs_wk	This table contains aggregate sales data for a Product, Location, Customer Segment and Week. The SLS_PR columns represent the metrics for that week that were on promotion, while the other metrics represent the sales metrics while the item was not on promotion.
rse_sls_pr_wk_a	This table contains aggregate sales data for the dimensions of a product and a week.
rse_sls_ph_lc_wk_a	This table contains aggregate sales transaction data for different product hierarchy/levels, at the store location/week dimension.
rse_sls_pr_lc_wk	This table contains aggregate sales data for a Product, Location, and Week. The SLS_PR columns represent the metrics for that week that were on promotion, while the other metrics represent the sales metrics while the item was not on promotion.
rse_sls_txn	This table contains sales transaction data.
rse_prod_attr	This is the table that holds product attributes.
rse_prod_attr_grp	This is the table used to load the associations of CM Groups to product attributes.
rse_prod_attr_grp_value	This is the table used to load the associations of CM Groups to product attributes and its values
rse_prod_attr_grp_value_map	This is the table used to load the associations of CM Groups to product attributes, group values and actual product attribute values
rse_like_loc	This is the table used to load the like stores for CMGroup or Category.
rse_hier_level	This table defines the various levels for all the hierarchies.

Table 21-4 (Cont.) Schema Objects

Table Name	Description
rse_hier_type	This table defines the available hierarchies for use within the RSE applications.
rse_fake_cust	Table for specifying customers who are considered as fake customers. A fake customer is a customer who purchases too many transactions to be considered a single customer. Examples are generic store cards.
rse_loc_src_xref	This table contains integration ID information that enables interaction with other systems, using IDs that other systems can accommodate for the Location Hierarchy.
rse_prod_src_xref	This table contains integration ID information that enables interaction with other systems, using IDs that other systems can accommodate for the Product Hierarchy.
rse_log_msg	This table contains messages logged while database, batch or business processing.
w_party_per_d	This table contains customer data and its attribute.
cis_cust_attr_vw	Customer Attributes - This view provides basic customer attributes.
cis_cust_trans_attr_vw	Customer Transaction Attributes - This view provides attributes for customer transactions.
cis_cust_trans_ph_attr_vw	Customer Transaction Attributes - This view provides product attributes for customer transactions.
cis_custseg_attr_exp_vw	This view provides an export of the attributes that define a segment.
cis_custseg_cat_attr_exp_vw	This view provides an export of the product attributes that define a segment.
cis_custseg_cust_exp_vw	This view provides the members for an exportable set of segments.
cis_custseg_exp_vw	This view provides an exportable set of clusters for customer segmentation.
cis_sls_cust_cal_a	This table contains aggregate customer sales data for a configurable level of the calendar hierarchy. The table is to be partitioned by Calendar and is also be suitable for sub partitioning by Customer using a Hash Partition strategy, so that subsequent uses can operate within the confines of a given Hash partition.
cis_sls_cust_ph_cal_a	This table contains aggregate customer sales data for a configurable level of the calendar hierarchy, for a selected set of product hierarchies. The table is to be partitioned by Calendar and is also be suitable for sub-partitioning by Customer using a Hash Partition strategy, so that subsequent uses can operate within the confines of a given Hash partition.
cis_sls_ph_a	This table contains aggregate sales data for all product hierarchy members of a configured hierarchy type and level. This can be used to identify the Top Categories for use by things like Customer Segmentation.
cis_cluster_set_exp_vw	This view provides an exportable set of clusters to send to Cat Man.
cis_store_cluster_exp_vw	This view provides an exportable set of clusters for stores.
cis_store_cluster_mem_exp_vw	This view provides the members with an exportable set of segments.
cis_store_cluster_prop_exp_vw	This view provides an exportable set of clusters for stores.
cis_cluster_sls_ph_lc_a	This table contains calendar level aggregates for the various clusters. The table is to be partitioned by Calendar.

Table 21-4 (Cont.) Schema Objects

Table Name	Description
cis_cluster_summ_level_attr	These are metrics generated at cluster/attribute/business object level. There are metrics that are generated at that level such as centroid.
cis_prod_attr_loc_share	This table contains aggregate sales data for product attribute values as well as the share that these values are with respect to the total sales for the product hierarchy, calendar, location. The share value is a configurable value, which can either be based on sales units, sales amount or profit amount.

Process Orchestration and Monitoring

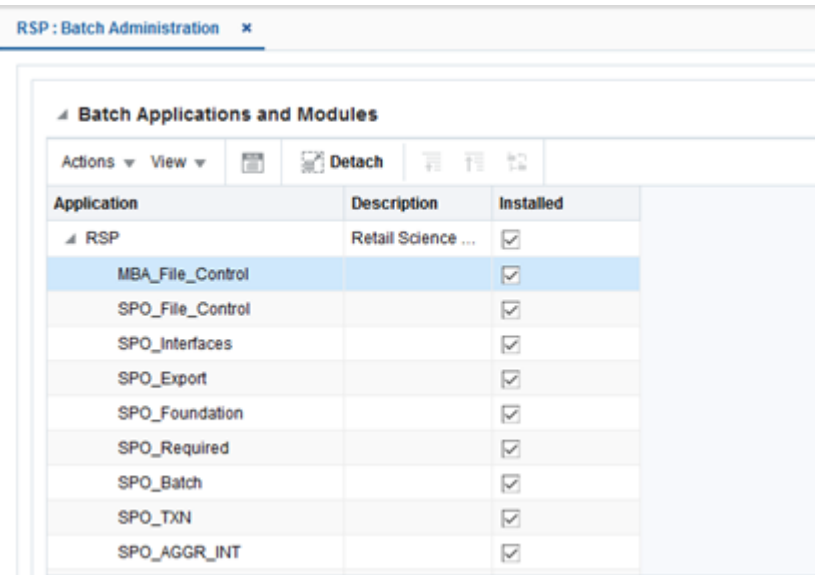
This chapter describes the basic workflow that triggers batch processes from Process Orchestration and Monitoring (POAM), including monitoring and error handling.

Batch Administration

Using the Batch Administration functionality, shown in [Figure 22-1](#), the user can enable and disable application-specific batch processes as a whole or enable and disable individual batch processes at the job level. Any modifications made will become effective with the next scheduler day load (except batch throttling configurations). The execution of application batch processes can be controlled by selecting or deselecting the check box.

To access Batch Administration, from the Tasks list, select Batch Administration.

Figure 22-1 Batch Administration



Batch Monitoring

Using Batch Monitoring, shown in [Figure 22-2](#), the user can monitor the current status of different batch processes at cycle level including nightly, ad hoc, and hourly. This screen displays a pie chart section that provides a summary view of the current cycle execution. A job can have 12 different job statuses: Loaded, Pending, Completed, Error, Held, Skipped, Running, Long Running, Disabled, System Held, Skipped on Error, and Waiting.

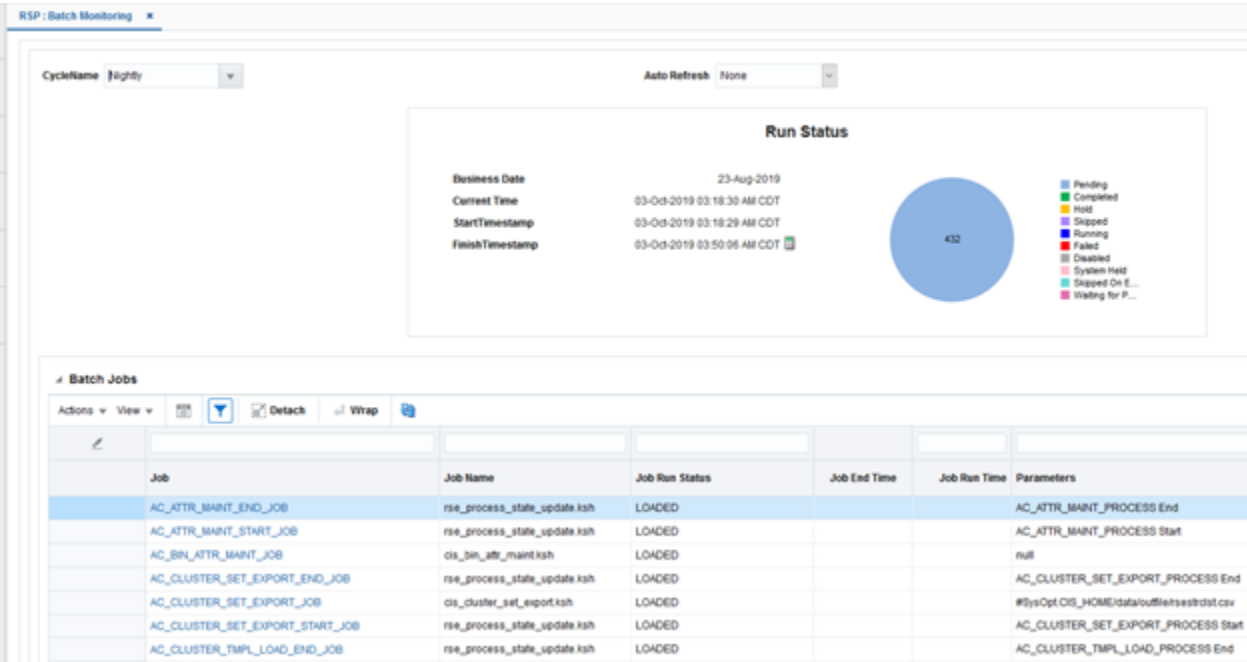
To access Batch Monitoring, from the Tasks List, select Batch Monitoring.

Monitoring a Batch Cycle

To create a new scheduler day.

- 1. Access Batch Monitoring.
- 2. Select the schedule tile for the application for which you want to create a new scheduler day.
- 3. Click **Create Schedule**. A popup window appears with the following: Do you want to create a new <application name> schedule for <POM business data>?
- 4. Click **Yes** to confirm. A new scheduler day will be created.
- 5. From the Monitoring Cycle list box, select the Nightly/Ad hoc/Hourly cycle. Note that the individual nightly batch job with its run status is loaded.
- 6. A pie chart displays the summary view of the current cycle execution.

Figure 22-2 Batch Monitoring



Triggering Process from POAM

Running a Nightly Batch Process

To run a nightly batch process via the POM UI, complete the following steps:

- 1. Log into POAM using a valid username and password.
- 2. Navigate to Batch Monitoring.
- 3. Select the **Nightly** tab located below the schedule tiles.
- 4. From the Auto Refresh drop-down list, select **30 sec**.

5. Click the Scheduler Tasks link.
6. In AIF applications Nightly Scheduler Task, select a Task ID, and select Actions > Edit. Alternatively, you can select the Pencil icon.
7. In Edit Task Nightly, do the following:
 - a. Set Enabled to true.
 - b. Set the Schedule Date.
 - c. Set the Schedule Time.
 - d. Click Ok.
8. The nightly batch will kick-off at the specified date and time.

Alternatively, running a nightly batch process can be done via Postman, by executing the following:

1. From Postman, do the following:
 - a. Post the url and the process to be started in the following format, replacing POAM_URL with the correct value.

POAM_URL/ProcessServices/services/private/executionEngine/schedules/RSP/execution

For example:

http://<pom-server-url>.us.oracle.com/ ProcessServices/services/private/executionEngine/schedules/RSP/execution.
 - b. In the Authorization section, provide the correct user name and password for the environment.
 - c. In the body, provide details for Cycle name, Flow name, and Request type. For example:

```
{  
  "cycleName" : "Nightly",  
  "flowName" : "Nightly",  
  "requestType" : "External Request"  
}
```
 - d. Click **Post**. The body indicates that the execution has started. The process begins executing in POAM.

Monitoring the Process Executions

After triggering the process from Postman, the user can view the progress of the batch execution and the state of the batch process (either pass or fail) in the POAM UI. In the POAM UI, open the Batch Monitoring screen and select **Nightly**. Refresh the scheduler day you created. The batch schedule status can be viewed in the displayed graph; if Auto Refresh is set, then the graph is automatically updated.

Note the following:

- The nightly batch schedule, once executed successfully, opens the next day automatically.
- If any batch failures have occurred, the next day will not open automatically. In such cases, the failure must be resolved, the existing scheduler day must be closed, and the new scheduler day must be opened manually.

- AIF Data and AIF Apps schedules must always be on same business dates. Triggering AIF DATA will automatically start the AIF application batch processes (once AIF Data completes, it starts the AIF Apps schedule.)

Error Handling

A batch process can fail because of the following:

- An actual batch process failure.
- A failure as a result of the loss of communication between POAM JOS components.
- The Error Source column lists the source of error. If the source is POAM, it is generally a connection issue.

If the source of the error is Job Admin, it can be either a connection failure or an actual job failure. This can be determined by examining the logs for JOS JOB ADMIN.

Figure 22-3 Error Handling

Job Run Status	Error Source	Job	Job Name	Job End Time	Job Run Time	Parameters
ERROR	Job Admin	ASO_INTRADAY_INPUT_FILES_WAIT_JOB	rse_batch_zip_file_wait.ksh	17-Jan-2020 04...	31	INTRADAY
LOADED		ASO_ASSORTMENT_FINALIZED_LOAD_JOB	so_assortment_finalized_load.ksh			null
LOADED		ASO_ASSORTMENT_FINALIZED_STG_JOB	so_assortment_finalized_stg.ksh			#SysOptASO_HOME\data\inf\assortme...

JOS JOB ADMIN

1. Log into JOS JOB ADMIN using a valid user name and password.
2. Navigate to System Logs.
3. Against the job name, there is a link to download the log. Also, clicking on the job name brings up a screen that also has a link to download the logs.

Skipping a Failed Job

If any job has failed for a known reason, the user can bypass it. In this way, the schedule releases the dependency and can continue executing other jobs.

To skip a failed job, complete the following steps:

1. Navigate to POAM > Batch Monitoring.
2. Select the job row that you want to skip.
3. Click **Skip or Release Skip** from More Actions.
4. Add any appropriate comments.
5. The status of the job will change to SKIPPED if the action is SKIP or to LOADED if the action is Release Skip.
6. Note that a job that already has a status of Skipped can only be released.

Targeted Items Recommendation

This chapter explains the steps needed in AIF to use the Targeted Items Recommendation (TIR) service from XStore or any other relevant app.

TIR can be primarily classified into four types:

1. Out-of-stock substitutions: similar items that are available/allocatable in the specific store/channel.
2. Up-sell: higher-margin, comparable alternatives, respecting price bands and customer context.
3. Cross-sell: complementary items using item-item and basket affinity across categories.
4. Repetition/replenishment: based on individual or cohort purchase cadence and preferences.

Each of these four TIR types involve specific kinds of ML models and approaches to recommend appropriate items. The recommendations may also vary by stores and customer segments.

Attribute Similarity Based TIR

This section explains the science, workflow steps, data requirements, and validation checks to use the Attribute Similarity based TIR. The science is the same as the Attribute Similarity calculation in the Demand Transference (DT) module.

Currently, only the Up-sell feature is supported using this approach.

Attribute Similarity Science

The idea is to compare products based on their similarities calculated from product attributes in order to determine what, if any, products customers might buy if the product they want to buy is for some reason unavailable. In this way, planning, ordering, and other business decisions can be optimized.

Role of Attributes in Calculating Similarities

The similarity of two SKUs is based in part on how many attribute values they have in common (the more in common, the higher the similarity of the two SKUs). Because the attributes play such an important role in calculating similarities, attribute quality is important when performing the calculation. The similarities calculated are only at the Customer Segment-Chain and Location-Company level.

Attribute Data Requirements

The attribute values for the similarity calculation must meet the following requirements:

Set of Attributes: Each product category is characterized by a unique set of attributes. These attributes differ from category to category. For example, for yogurt, the attributes might be size, flavor, brand, fat percentage, and pack size. For chocolate, the attributes might be size, brand, milk/dark, nut type, and package type. Two categories can both have brand, but the brand

attribute will have different values for each of the categories. So, brand is actually a different attribute for each category.

Mapping: Each item in the category must be mapped to its set of attribute values. This information must be obtained from the retailer. Null values are acceptable as long as they are not too numerous. The calculation can still run even if some attribute values are listed as null for some items in a category, but too many null values decrease the reliability of the generated similarities. In particular, too many SKU pairs may come out as less similar than they should be. Null values have a particular use in accommodating categories that are actually a union of more than one category.

Significance: The attributes for a category must be the ones that the customers actually pay attention to when shopping in the category. They are attributes that actually affect the customers' purchasing decisions. Note that the similarity calculation will still complete even with attributes that do not affect customer behavior, but the similarities produced will be less distinguishing. For example, a category has a Supplier attribute, which indicates for a given product which supplier shipped an item to the grocer. This attribute may be important to the grocer for accurate bookkeeping, but it has no effect on the customer's purchasing behavior because it is not reflected in the item itself nor is it something that the customer is concerned about. However, if it is included when setting up attributes, then the effect would be to increase the similarity of items that were from the same supplier. This is a false similarity, since it does not reflect how the customer actually views these items. In particular, if the supplier is a duplicate of the Brand attribute, then the similarity of products within the same Brand would be unintentionally increased. The process of obtaining attributes for a category and performing a mapping of items in the category to attribute values is likely to require a significant amount of time and labor, even if the retailer has the information available, since this must be done for every category.

Guidelines on Number of Attributes and Attribute Values

The number of attributes and attribute values must be enough to distinguish the SKUs within a category. That is, for a given set of attribute values, the number of SKUs in the category all having those values must be a small number. A maximum of seven SKUs is recommended. For example, the Cookie category at a grocer has only three attributes, Brand, Package Size, and Organic. If Brand has seven values, Package Size has three values, and Organic has two values (either Yes or No), then the total number of combinations of attribute values is $7 \times 3 \times 2 = 42$. For 600 different cookie SKUs, the average for each combination of attribute values will represent $600 / 42 =$ approximately 14 different SKUs. The distribution of SKUs among the 42 different sets of values will not be an even 14, as some sets of values will have much more than 14, while others will have less. The three attributes alone are not enough to provide enough distinguishing power among the cookie SKUs. If Flavor is an important determinant of customer purchases, it should be added to the Cookie category.

The guideline of a maximum of seven indicates that additional attributes are necessary. It is worth examining those sets of attribute values that have the largest number of SKUs associated with them in order to see what attributes can be added to reduce the number of SKUs. It is not just the number of attributes that is important, but how many values each attribute has. For example, if Brand had 100 values instead of seven values, then the total number of attribute-value combinations is $100 \times 3 \times 2 = 600$. It might seem that an easy way to achieve the maximum of seven is to expand the number of values in the attributes. However, this results in each SKU being similar to only a small number of other SKUs. For a single attribute for cookies with 600 different values, it might then be possible to assign one value to each cookie SKU, separating all of the 600 cookies SKUs with a single attribute. However, this would make each cookie SKU completely dissimilar (similarity of 0) from all other cookie SKUs. Putting all 600 SKUs each into a separate Brand causes a complete loss of any similarity information among the SKUs.

For an opposite example, consider 11 attributes, each with only two values. There is a total of $2^{11} = 2,048$ combinations of values, so that may be enough to encode 2,000 SKUs, even though there are only $2 \times 11 = 22$ distinct attribute values over the 11 attributes. In general, having more attributes is better, and it is better to increase the number of attributes rather than increase the number of attribute values of a single attribute. However, this is not always possible, and it is better to have the attribute with many values than not have the attribute at all. Flavor, for example, can have many values, as can Color. The more SKUs in the category, the more attributes and attribute values will be needed to achieve the maximum of seven.

Effect on Similarity Values

Suppose the set A of SKUs consists of 22 SKUs, all with the same attribute values, and the set B of SKUs consists of 25 SKUs, all with the same attribute values (but a different set of attribute values from set A). If the set A consists of cookie SKUs all with a package size of Small, and set B consists of cookie SKUs with the same attribute values as A except the size is Medium, then every SKU in A has a similarity of 1 to every other SKU in A, and every SKU in A is similar to at least 22 other SKUs. Every SKU in A is similar to every SKU in B, since they only differ in one attribute value (namely size). So, a SKU in set A is similar to at least $21 + 25 = 46$ SKUs. It is possible that a SKU in A being similar to 46 other SKUs in fact represents reality, but if it does not, then using additional attributes that distinguish the SKUs in A and in B will reduce the number of similar SKUs.

Avoiding Attributes with Many Values

Attributes with a large number of values occur frequently. For example, a color attribute in any clothing category might have several shades of each color. Midnight blue, sea blue, and sky blue may all be separate attribute values of the Color attribute; the problem is that in the similarity calculation, a midnight blue item and a sea blue item would be considered completely dissimilar colors, because these two color attribute values are different; in reality, because they are both shades of blue, they should be somewhat similar. One solution is to split the color attribute into two separate attributes, a primary color attribute and a modifier. In this example, the primary color would be blue and the modifiers midnight, sea, and sky.

Functional-Fit Attributes

A functional fit attribute is one where there is no substitution across the attribute's values. For example, batteries of different sizes cannot be substituted for one another. Any category where size determines the functional suitability of the item will have size as a functional-fit attribute.

The functional-fit attributes are used to set the similarity of two SKUs to be zero if the SKUs differ in any functional-fit attribute. Without the functional-fit information, the two SKUs may have non-zero similarity, and that would be erroneous, such as batteries of different sizes.

Designating an attribute as functional fit can also be useful any time the attribute is unlikely to have substitution across it (for example, caffeinated versus decaffeinated coffee). This is not exactly functional fit; however, substitution is unlikely, so it is better to mark the attribute as functional fit.

One approach to avoiding having to define large numbers of attributes and attribute values is to use functional-fit attributes. This approach does not help achieve the maximum of seven, but it can help decrease the number of SKUs that are similar to a given SKU. For example, with the sets A and B of cookie SKUs, if size were designated as functional fit, then the similarity between SKUs in A and SKUs in B would become zero. However, that designating size as functional fit does nothing about the 22 SKUs in A that all have a similarity of 1 to each other, since their attribute values are all the same. (Similar comments apply to set B.)

If the attributes and attribute values are insufficient to reach the maximum of seven SKUs per set of attribute values, functional-fit attributes can be used to decrease the number of SKUs

that will share a positive similarity. This is a second-best approach, and it is better to design a proper set of attributes and attribute values, in order to:

- Achieve the maximum of seven SKUs.
- Provide positive similarity between SKUs that should have a positive similarity. Using functional-fit attributes reduces the number of SKUs that will share a positive similarity, but it may reduce it too much and remove similarities from pairs of SKUs that should have a positive similarity. For example, in the sets A and B, the similarity between a SKU in A and a SKU in B becomes zero, which does not reflect reality since the SKUs in A and in B share common attribute values except for size.
- Keep the second-best approach as a last resort, in case time is insufficient for designing a good set of attributes for a category.

Using Null as an Attribute Value

Null is a legitimate attribute value for an item to have; it denotes either of the following:

- The attribute value is unknown. In this case, the item has a value for this attribute, but the value is unknown. Attribute data for a category is rarely perfect, it is entirely possible that attribute values for some items were never recorded or are known to be incorrect. The use of null because of imperfections in the attribute data must only be for those cases where it is truly necessary.
- The attribute does not apply to the item. Not all attributes in a category apply to all items, especially when the category consists of sub-categories. The ideal is to define the items in the category so that every attribute for the category applies to every item, but this may not be possible. This usually occurs when an attribute applies to an item only if the item has a particular value for another attribute. For example, in the meat snacks category, the attribute grass fed only applies if the item has beef for the product type attribute. If the item is turkey, then the grass-fed attribute must be set to null for the item. The grass-fed attribute must be true or false only for beef products.

Using null is not the ideal way of handling the second case listed above. If possible, it is preferable to split the items in the category into separate categories. For example, this means putting beef products into their own category, so that the grass-fed attribute applies only to this new category and can be removed from the non-beef products. This is only reasonable if the beef products are not substitutes for the other products in the meat snacks category (which is probably not true since the customer might reasonably substitute turkey jerky for beef jerky). In the case of meat snacks, the best solution is to use null for the grass-fed attribute for non-beef products.

Effect of Null Attribute Values on Similarity Values

A null attribute value for an item means that the item does not match any other item in that attribute, even if the other item also has null for the same attribute. Null as the value of an attribute makes the item less similar to every other item, including items that also have null for the same attribute. If an item has null for every attribute, then this item has a similarity of zero to every other item, regardless of what values the other items have for their attributes. A value of null for an attribute is thus treated as special in similarity calculations. It is recommended to keep the use of null to a minimum in order to avoid spuriously indicating that items are dissimilar when they are actually similar.

If the intention is to have null for an attribute value, it is essential to actually use null and not create a string such as "none" to use where null is actually meant. Designating a string such as "none" to use in place of null actually does the opposite of what is described above, because two items with a value of "none" for the same attribute will now be considered similar in that attribute since they have the same value for the attribute. This is directly opposite to the intention of null, as described above.

Categories Containing Sub-Categories

It is possible for a category to consist of items from several categories. As it may be impractical to separate the category into its component categories, it is necessary to create attributes for the category that can approximate the separation. The following process describes how to do this, using the category Oral Care. This category consists of toothbrushes, toothpaste, dental floss, and mouth wash; it is really four separate categories whose items have all been classified together into a single category.

1. Create one attribute that can separate the category into its component categories. For example, for Oral Care, create an attribute called Product Type that has four values: "toothbrush," "toothpaste," "floss," and "mouthwash." The values of this separating attribute should describe the function of the product. It should separate the category into groups of items in which the items within each group are unlikely to substitute for items in another group. For example, toothbrushes are unlikely to substitute for toothpaste. Constructing a separating attribute is not always as simple as this, as discussed below.
2. Designate this separating attribute as functional fit. This means that there will be no similarity between items that are in the different groups specified by the separating attribute. In this case, there is no similarity between the four component groups of Oral Care.
3. Create a single set of attributes suitable for all of the component categories. This typically involves taking the union of the attributes that describe each component category separately. For example, the set of attributes that describe toothpaste are different than the set of attributes that describe toothbrushes, but in this scheme, just take the union of both sets of attributes. Toothbrushes have an attribute called "Bristle hardness," which does not apply to toothpaste, and likewise toothpaste has a flavor attribute that does not apply to toothbrushes. Nonetheless, take all of the attributes together, including the ones for floss and mouthwash. It is possible that an attribute can apply across several of the component categories. For example, only a single flavor attribute is necessary, instead of having three attributes called Floss-Flavor, Toothpaste-Flavor, and Mouthwash-Flavor. Either approach is correct, but having three separate flavor attributes may require more work to create.
4. Assign null values appropriately. For example, for attributes that apply only to toothpaste, set them to null for any toothbrush. Likewise, for attributes that apply only to toothbrushes, set them to null for any toothpaste.

Here is an example of the Oral Care category and how to set null for its attributes. Assume a separating attribute for Oral Care as described above, and assume, in addition, the following attributes: Thickness, Length, Contains Alcohol, and Flavor. (In reality, there would be more.) The following table shows which attributes apply to which products. Null indicates that the attribute does not apply to the product, and therefore, should be set to null for the product.

Table 23-1 Oral Care Category Example

Product Type	Thickness	Length	Contains Alcohol	Flavor
Floss	Applies	Applies	Null	Applies
Mouthwash	Null	Null	Applies	Applies
Toothpaste	Null	Null	Null	Applies
Toothbrush	Null	Null	Null	Null

In Step 1, you may not find a separating attribute. Instead, you may need to use an attribute that separates the items into groups but in which some of the groups may have items that can substitute for each other. For example, consider a category that consists of Shampoo, Shampoo and Conditioner, and Conditioner, where Shampoo and Conditioner are products

that have shampoo and conditioner in one bottle. In this case, a separating attribute has three values, Shampoo, Shampoo and Conditioner, and Conditioner. If this attribute is set as functional fit, there is no similarity between Shampoo and Shampoo and Conditioner. In this example, the approximation may be good enough, since although substitution can take place between Shampoo and Shampoo and Conditioner, it is probably not large.

Setting Up Categories

In general, a category is a set of items that are substitutable with each other (if there are no functional-fit attributes). The categories at a retailer can all be derived by choosing the correct level of the merchandise hierarchy at the retailer. The configuration supports choosing which level of the merchandise hierarchy is to be used as the category level.

Similarity calculations only occur within the category, since the categories define the sets of items that substitute each other.

Frequently, retailers will have categories that are actually unions of categories. For example, a retailer might have a Hair Care category that contains shampoo, conditioner, and hair oil. The retailer may not want to separate out this category into three separate categories of Shampoo, Conditioner, and Hair Oil, if, for example, a single person in the organization is responsible for all three. The problem is that these three types of products do not share a common set of attributes. The attributes describing Hair Oil are not the same ones needed for describing Shampoo or Conditioner. These types of products may share common attributes, such as Scent, but each type of product also needs its own set of attributes. The solution is to define, in addition to the common attributes, a set of attributes for each product type. If an attribute applies only to Shampoo, and not to Conditioner or Hair Oil, then Conditioner SKUs and Hair Oil SKUs should have Null for the value of that attribute. This is a common use of null attribute values and makes it possible to handle the case of a category that is really the union of smaller sub-categories.

Historical Sales Data

Attribute Similarity calculation requires SKU-store-week sales-units aggregates. Typically, the data should not be aggregated to a higher location level than store because different stores usually have different assortments. Some atypical cases can occur in which aggregation across some stores is legitimate because the assortments are the same or nearly so, but this is generally not the case.

In addition to the SKU-store-week sales-units aggregates, it also requires promotion data. A flag indicates which SKU-store-weeks contain major promotions (ones that caused a very large increase in sales units, such as three times normal). It uses the promotion data to flatten the promotional spikes in the SKU-store-week sales-units aggregates. The flattened data is called the baseline sales. An alternative to flattening the promotions is to ignore the SKU-store-weeks where promotions occurred. However, to implement this, it is necessary to ignore all SKU-store-weeks in any week where a SKU-store was promoted, because it is not clear what the effect is on the other items when one SKU is promoted and the others are not. Removing that many SKU-store-weeks can leave little data remaining, especially since many retailers promote quite frequently. For this reason, it is better to flatten promotions and keep more data instead.

Similarly to the way in which promotions are handled, the calculation of the baseline sales also involves removing short-term downward spikes that are due to very short-term stock outs. Note that inventory information is not an input, and so the algorithm finds large but short-lasting dips in weekly sales units and fills those in. This handling of out-of-stock is not related to long-term out-of-stock conditions.

Loading Product Attribute Data into AIF

The following table shows the relevant parameters from RSE_CONFIG in the Manage System Configurations screen.

Table 23-2 Relevant Parameters from RSE_CONFIG

APPL_CODE	PARAM_NAME	PARAM_VALUE	DESCR
RSE	CMGRP_LEVEL_ID	Default value is 5.	The hierarchy level id which contains the level of the product hierarchy where the Category Management (CM) Group level exists, that is, the level at which the product attribute groups are loaded into AIF.
PMO	PMO_PROD_HIER_TYP	Default value is 3.	The hierarchy type id to use for the product. Keep the value at 3 (if Extended Merchandise Hierarchy has been loaded into AIF) or change to 1 (otherwise, for Normal Merchandise Hierarchy).

The following table shows the relevant jobs to load product attribute data into AIF. The same can also be achieved by the RSE_MASTER_ADHOC_AIF_JOB (rse_master.ksh) with parameter -P (process Product Attribute specific steps) or -A (process all AIF steps).

Table 23-3 Relevant Jobs to Load Attribute Data

JobName	Description	RmsBatch
RSE_CDA_ETL_LOAD_PROD_JOB	Populates the RSE_PROD_ATTR and RSE_ATTR_DESCR tables with product attributes pulled from the W_PRODUCT_D and W_RTL_ITEM_GRP1_D tables.	rse_cda_etl_load_product.ksh
RSE_PROD_ATTR_GRP_VALUE_STG_JOB	Stages associations of product attribute groups to product attributes from rse_prod_attr_grp_value_stg.txt	rse_prod_attr_grp_value_stg.ksh
RSE_PROD_ATTR_GRP_VALUE_COPY_JOB	Copies rse_prod_attr_grp_value_stg.txt file previously extracted from a ZIP to an internal location so it can be loaded through an external table.	
RSE_PROD_ATTR_GRP_VALUE_STG_CNE_JOB	Stages associations of product attribute groups to product attributes from rse_prod_attr_grp_value_stg.txt file into the RSE_PROD_ATTR_GRP_VALUE_STG table through an external table.	
RSE_PROD_ATTR_GRP_VALUE_LOAD_JOB	Populates the RSE_PROD_ATTR_GRP and RSE_PROD_ATTR_GRP_VALUE tables with product attribute group data pulled from the RSE_PROD_ATTR_GRP_VALUE_STG table.	rse_prod_attr_grp_value_load.ksh

Table 23-3 (Cont.) Relevant Jobs to Load Attribute Data

JobName	Description	RmsBatch
RSE_PROD_ATTR_VALUE_XREF_STG_JOB	Stages cross references of product attribute values to product attribute groups from rse_prod_attr_value_xref_stg.txt file into the RSE_PROD_ATTR_VALUE_XREF_STG table through SQL Loader.	rse_prod_attr_value_xref_stg.ksh
RSE_PROD_ATTR_VALUE_XREF_COPY_JOB	Copies rse_prod_attr_value_xref_stg.txt file previously extracted from a ZIP to an internal location so it can be loaded through an external table.	
RSE_PROD_ATTR_VALUE_XREF_STG_CNE_JOB	Stages cross references of product attribute values to product attribute groups from rse_prod_attr_value_xref_stg.txt file into the RSE_PROD_ATTR_VALUE_XREF_STG table through an external table.	
RSE_PROD_ATTR_VALUE_XREF_LOAD_JOB	Populates the RSE_PROD_ATTR_GRP_VALUE_MAP table with cross references of product attribute values to product attribute groups pulled from the RSE_PROD_ATTR_GRP_VALUE/RSE_PROD_ATTR_VALUE_XREF_STG tables.	rse_prod_attr_value_xref_load.ksh
RSE_CDA_ETL_LOAD_PROD_ADHOC_JOB	Ad hoc job that populates the RSE_PROD_ATTR and RSE_ATTR_DESCR tables with product attributes pulled from the W_PRODUCT_D and W_RTL_ITEM_GRP1_D tables.	rse_cda_etl_load_product.ksh
RSE_PROD_ATTR_GRP_VALUE_COPY_ADHOC_JOB	Ad hoc job that copies rse_prod_attr_grp_value_stg.txt file previously extracted from a ZIP to an internal location so it can be loaded through an external table.	
RSE_PROD_ATTR_GRP_VALUE_STG_CNE_ADHOC_JOB	Ad hoc job that stages associations of product attribute groups to product attributes from rse_prod_attr_grp_value_stg.txt file into the RSE_PROD_ATTR_GRP_VALUE_STG table through an external table.	
RSE_PROD_ATTR_GRP_VALUE_LOAD_ADHOC_JOB	Ad hoc job that populates the RSE_PROD_ATTR_GRP and RSE_PROD_ATTR_GRP_VALUE tables with product attribute group data pulled from the RSE_PROD_ATTR_GRP_VALUE_STG table.	rse_prod_attr_grp_value_load.ksh
RSE_PROD_ATTR_VALUE_XREF_COPY_ADHOC_JOB	Ad hoc job that copies rse_prod_attr_value_xref_stg.txt file previously extracted from a ZIP to an internal location so it can be loaded through an external table.	

Table 23-3 (Cont.) Relevant Jobs to Load Attribute Data

JobName	Description	RmsBatch
RSE_PROD_ATTR_VALUE_XREF_STG_CNE_ADHOC_JOB	Ad hoc job that stages cross references of product attribute values to product attribute groups from rse_prod_attr_value_xref_stg.txt file into the RSE_PROD_ATTR_VALUE_XREF_STG table through an external table.	
RSE_PROD_ATTR_VALUE_XREF_LOAD_ADHOC_JOB	Ad hoc job that populates the RSE_PROD_ATTR_GRP_VALUE_MAP table with cross references of product attribute values to product attribute groups pulled from the RSE_PROD_ATTR_GRP_VALUE/RSE_PROD_ATTR_VALUE_XREF_STG tables.	rse_prod_attr_value_xref_load.ksh

Table 23-4 Relevant Queries for Data Validation

```
-- query 1: gives the BUSINESS_OBJECT_MD_ID for product data
select * from RSE_BUSINESS_OBJECT_MD where NAME = 'PRODUCT';

-- query 2: gives the BUSINESS_OBJECT_DB_SRC_ID for product attribute groups
select * from RSE_BUSINESS_OBJECT_DB_SRC where DB_OBJECT_NAME =
'W_RTL_ITEM_GRP1_D';

-- query 3: gives the BUSINESS_OBJECT_ATTR_MD_ID based on BUSINESS_OBJECT_MD_ID and
BUSINESS_OBJECT_DB_SRC_ID
select * from RSE_BUSINESS_OBJECT_ATTR_MD where BUSINESS_OBJECT_MD_ID = <<ID from
query 1>> and BUSINESS_OBJECT_DB_SRC_ID = <<ID from query 2>> and EXCLUDE_FLG = 'N';

--query 4: gives the PROD_ATTR_GRP_ID based on BUSINESS_OBJECT_ATTR_MD_ID,
HIER_TYPE_ID, and PROD_HIER_ID (note that the PROD_HIER_ID in this table is at the level of
CMGRP_LEVEL_ID in RSE_CONFIG)
select * from RSE_PROD_ATTR_GRP where BUSINESS_OBJECT_ATTR_MD_ID = <<ID from query
3>> and HIER_TYPE_ID = <<PMO_PROD_HIER_TYPE from RSE_CONFIG>> and PROD_HIER_ID =
<<any particular product ID>> and CURRENT_FLG = 'Y' and DELETE_FLG = 'N';

--query 5: gives the PROD_ATTR_GRP_VALUE_ID based on PROD_ATTR_GRP_ID
select * from RSE_PROD_ATTR_GRP_VALUE where PROD_ATTR_GRP_ID = <<ID from query 4>>
and CURRENT_FLG = 'Y' and DELETE_FLG = 'N';

--query 6: gives the ATTR_VALUE_EXT_CODE based on PROD_ATTR_GRP_VALUE_ID
select * from RSE_PROD_ATTR_GRP_VALUE_MAP where PROD_ATTR_GRP_VALUE_ID = <<ID
from query 5>>;

--query 7: gives the ATTR_DESCR based on BUSINESS_OBJECT_ATTR_MD_ID (BOAM_ID) and
ATTR_VALUE_EXT_CODE
select * from RSE_ATTR_DESCR where BOAM_ID = <<ID from query 3>> and
ATTR_VALUE_EXT_CODE = <<ATTR_VALUE_EXT_CODE from query 6>>;
```

Table 23-4 (Cont.) Relevant Queries for Data Validation

```
--query 8: gives the leaf-level PROD_HIER_ID (SKUs) based on BUSINESS_OBJECT_ATTR_MD_ID,
ATTR_VALUE_EXT_CODE, and having the same PARENT_PROD_HIER_ID at the level of
CMGRP_LEVEL_ID
select * from RSE_PROD_ATTR where BUSINESS_OBJECT_ATTR_MD_ID = <<ID from query 3>>
and ATTR_VALUE_EXT_CODE = <<ATTR_VALUE_EXT_CODE from query 6>> and PROD_HIER_ID
in (select CHILD_PROD_HIER_ID
from RSE_PROD_HIER_TC where HIER_TYPE_ID = <<PMO_PROD_HIER_TYPE from
RSE_CONFIG>> and PARENT_HIER_LEVEL_ID = <<CMGRP_LEVEL_ID from RSE_CONFIG>> and
PARENT_PROD_HIER_ID = <<PROD_HIER_ID from query 4>> and CURRENT_FLG = 'Y');

--query 9: checking the HIER_LEVEL_ID for the relevant product
select * from RSE_PROD_HIER where HIER_TYPE_ID = <<PMO_PROD_HIER_TYPE from
RSE_CONFIG>> and ID = <<PROD_HIER_ID from query 4 or query 8>> and CURRENT_FLG = 'Y'
and DELETE_FLG = 'N';
```

Functional-Fit Attributes are very important for TIR. Size may be a functional-fit attribute, but Color may not be. Whether an attribute is functional-fit or not may depend on the specific business needs of the retailer. The idea is that if an attribute is functional-fit then an item should not be recommended if the value of the functional-fit attribute does not match that of the main item. For example, if Size is considered a functional-fit attribute, then the recommended item's Size must match that of the main item. Functional-Fit Attributes must be loaded as FUNC_ATTR_FLG = 'Y' into the table RSE_PROD_ATTR_GRP by using the table RSE_PROD_ATTR_GRP_VALUE_STG. All other (non-functional-fit) attributes must be loaded as FUNC_ATTR_FLG = 'N'.

Calculating Product-Location Range Data and Baseline Sales in AIF

The following table shows the relevant parameters from RSE_CONFIG in the Manage System Configurations screen.

Table 23-5 Relevant Parameters from RSE_CONFIG

APPL_CODE	PARAM_NAME	PARAM_VALUE	DESCR
RSE	RSE_PROD_LOC_STATUS_SOURCE	Default value is CALC.	The source of data for product location ranging data (RI or CALC).
RSE	IMPLY_OUT_OF_RANGE_AFTER	Default value is 8.	The number of weeks with no sales required before a SKU is considered to be out of range for a location.

The following table shows relevant jobs to load or calculate product-location range data and calculate baseline sales in AIF.

Table 23-6 Relevant Jobs to Load or Calculate

JobName	Description	RmsBatch
RSE_PROD_LOC_RANGE_SETUP_JOB	Sets up a queue for the RSE_PROD_LOC_RANGE service that populates the RSE_PROD_LOC_STATUS table with product-location range data using data in the W_RTL_IT_LC_D/ RSE_SLS_PR_LC_WK tables. Default parameters for this job are (-n 2 -f Y).	rse_prod_loc_range_setu p.ksh
RSE_PROD_LOC_RANGE_PROCES S_JOB	Processes a queue for the RSE_PROD_LOC_RANGE service that populates the RSE_PROD_LOC_STATUS table with product-location range data using data in the W_RTL_IT_LC_D/ RSE_SLS_PR_LC_WK tables.	rse_prod_loc_range_proc ess.ksh
RSE_BASELINE_SETUP_JOB	Sets up a queue for the RSE_BASELINE service that executes baseline sales calculation using data in the RSE_SLS_PR_LC_CS_WK/ RSE_PROD_LOC_STATUS tables and populates the RSE_BL_SLS_PR_LC_CS_WK table.	rse_baseline_setup.ksh
RSE_BASELINE_PROCESS_JOB	Processes a queue for the RSE_BASELINE service that executes baseline sales calculation using data in the RSE_SLS_PR_LC_CS_WK/ RSE_PROD_LOC_STATUS tables and populates the RSE_BL_SLS_PR_LC_CS_WK table.	rse_baseline_process.ks h
RSE_PROD_LOC_RANGE_SETUP _ADHOC_JOB	Ad hoc job that sets up a queue for the RSE_PROD_LOC_RANGE service that populates the RSE_PROD_LOC_STATUS table with product-location range data using data in the W_RTL_IT_LC_D/ RSE_SLS_PR_LC_WK tables. Default parameters for this job are (-n 2 -f Y).	rse_prod_loc_range_setu p.ksh
RSE_PROD_LOC_RANGE_PROCES S_ADHOC_JOB	Ad hoc job that processes a queue for the RSE_PROD_LOC_RANGE service that populates the RSE_PROD_LOC_STATUS table with product-location range data using data in the W_RTL_IT_LC_D/ RSE_SLS_PR_LC_WK tables.	rse_prod_loc_range_proc ess.ksh

Table 23-6 (Cont.) Relevant Jobs to Load or Calculate

JobName	Description	RmsBatch
RSE_BASELINE_SETUP_ADHOC_JOB	Ad hoc job that sets up a queue for the RSE_BASELINE service that executes baseline sales calculation using data in the RSE_SLS_PR_LC_CS_WK/ RSE_PROD_LOC_STATUS tables and populates the RSE_BL_SLS_PR_LC_CS_WK table.	rse_baseline_setup.ksh
RSE_BASELINE_PROCESS_ADHOC_JOB	Ad hoc job that processes a queue for the RSE_BASELINE service that executes baseline sales calculation using data in the RSE_SLS_PR_LC_CS_WK/ RSE_PROD_LOC_STATUS tables and populates the RSE_BL_SLS_PR_LC_CS_WK table.	rse_baseline_process.ksh

Table 23-7 Relevant Queries for Data Validation

```
--query 10: checking the presence of a particular SKU-store in the product-location range data
select * from RSE_PROD_LOC_STATUS where PROD_HIER_ID = <<leaf level PROD_HIER_ID>> and
LOC_HIER_ID = <<leaf level LOC_HIER_ID>>;

--query 11: checking the last sales of a particular SKU-Store
select * from RSE_PROD_LOC_LAST_SLS where PROD_HIER_ID = <<leaf level PROD_HIER_ID>>
and LOC_HIER_ID = <<leaf level LOC_HIER_ID>>;

--query 12: getting the CHAIN customer segment and all other child segments
select * from RSE_CUSTSEG_HIER order by ID;

--query 13: checking the presence of a particular SKU-store in the baseline sales data
select * from RSE_BL_SLS_PR_LC_CS_WK where PROD_HIER_ID = <<leaf level PROD_HIER_ID>>
and LOC_HIER_ID = <<leaf level LOC_HIER_ID>> order by CUSTSEG_HIER_ID, CAL_HIER_ID;

--query 14: getting the summary statistics of a particular SKU-store from the baseline sales data
select CUSTSEG_HIER_ID, min(CAL_HIER_ID), max(CAL_HIER_ID), sum(BL_SLS_QTY)
from RSE_BL_SLS_PR_LC_CS_WK
where PROD_HIER_ID = <<leaf level PROD_HIER_ID>> and LOC_HIER_ID = <<leaf level
LOC_HIER_ID>> group by CUSTSEG_HIER_ID order by CUSTSEG_HIER_ID;
```

Presence of a SKU-Store in the product-location range data and baseline sales tables are critical to calculate attribute similarities. If there is difficulty in populating the RSE_PROD_LOC_STATUS table with specific product-location records when RSE_PROD_LOC_STATUS_SOURCE = CALC in RSE_CONFIG, then try changing the values of IMPLY_OUT_OF_RANGE_AFTER in RSE_CONFIG and/or the parameter -n (number of latest historical weeks to process) in RSE_PROD_LOC_RANGE_SETUP_JOB/ RSE_PROD_LOC_RANGE_SETUP_ADHOC_JOB and then re-run the jobs to re-populate the table. For the population of RSE_BL_SLS_PR_LC_CS_WK with specific product-location records, those records must be present in RSE_PROD_LOC_STATUS and RSE_SLS_PR_LC_CS_WK.

Attribute Similarity Calculation as Part of AIF Forecasting

The expectation here is that the implementer is familiar with the AIF Forecasting module in the Manage Forecast Configurations screen. This subsection will go over the specific parameters and settings required for the attribute similarity calculation for TIR. The main goal is here is not to generate an accurate forecast, rather just generate the attribute similarities as part of the forecasting process.

The following table shows relevant parameters from RSE_CONFIG in the Manage System Configurations screen.

Table 23-8 Relevant Parameters from RSE_CONFIG

APPL_CODE	PARAM_NAME	PARAM_VALUE	DESCR
PMO	MAX_NUM_WEEKS_FOR_SIM_MDL_SLS	Default value is 104.	The maximum number of weeks which should be used during Similarity model sales calculation.
PMO	PMO_SIM_MIN_REQ_LC_SLS_WKS	Default value is 10.	The minimum number of weeks per location required for Similarity model build.
PMO	PMO_SIM_MIN_REQ_PR_LC_SLS_WKS	Default value is 10.	The minimum number of weeks per product-location required for Similarity model build.
PMO	SIM_CALC_INT_LENGTH	Default value is 8.	The number of weeks to group together in an interval for the Similarity, Attribute Weight calculation.

1. In the Manage System Configurations screen, set the USED_BY_FCST_APP_FLG value to Y for the TIR row in the RSE_APP_SOURCE table.
2. In the Setup train stop of the Manage Forecast Configurations screen, create a new forecast run type with the following parameters:
 - Data Source: Store Sales
 - Forecast Method: Sales and Promo or Life Cycle
 - Forecast Measure: Any available option
 - Forecast by Customer Segments: Turned Off
 - Forecast by Price Zone: Turned Off
 - Hierarchy Type - Merchandise: Extended Merchandise Hierarchy (if PMO_PROD_HIER_TYPE = 3 in RSE_CONFIG) or Normal Merchandise Hierarchy (if PMO_PROD_HIER_TYPE = 1 in RSE_CONFIG)
 - Hierarchy Type - Location: Normal Location Hierarchy or any Alternate Location Hierarchy
 - Hierarchy Type - Calendar: Fiscal Calendar Hierarchy
 - Forecast Level - Merchandise: Style Color Size (if using Extended Merchandise Hierarchy) or SKU (if using Normal Merchandise Hierarchy)

- Forecast Level - Location: Location (if using Normal Location Hierarchy) or the Leaf Level (if using Alternate Location Hierarchy)
- Forecast Level - Calendar: Week

If a run type already exists with these configurations, then that run type could also be potentially used for TIR and there is no need to create a new one in that scenario. After the run type is created, data aggregation for historical sales in PMO_ACTIVITIES and historical cumulative sales in PMO_CUM_SLS (this table is only needed for the Life Cycle Forecast Method) needs to be done from the UI, if the respective AGGR_LVL_KEY does not have the aggregated data.

Before using the Extended Merchandise Hierarchy (RSE_PROD_SRC_XREF, RSE_PROD_HIER, RSE_PROD_HIER_TC, RSE_PROD_HIER_DH) and/or an Alternate Location Hierarchy (RSE_LOC_SRC_XREF, RSE_LOC_HIER, RSE_LOC_HIER_TC, RSE_LOC_HIER_DH), the respective hierarchy tables in AIF must be verified to ensure that proper and complete hierarchy data exists.

3. In the Map train stop of the Manage Forecast Configurations screen, map the run type to the TIR app. At any point in time, at most one run type should be mapped to TIR. If another run type is mapped to TIR, then it is important to delete the mapping of the previous run type from TIR. Otherwise, among the run types that are mapped to TIR, the one with the greatest run type ID will be used.
4. In the Test train stop of the Manage Forecast Configurations screen, select the TIR run type row in the top table. In the bottom table, create a new run by opening the 'Scope' tab and selecting the radio button 'Run estimation only' or 'Run estimation and base demand' and optionally turning ON the 'Run Forecast' toggle button. Note that for the purpose of TIR, only creating an estimation run is sufficient. The base demand and forecast steps are optional. The run should be set up with appropriate parameters for a Sales and Promo or Life Cycle execution. While creating the run, make the following choices under the 'New Item Forecast Method' section of the 'New Item Fcst' tab:
 - Radio button selection: Calculate like items and generate forecast for new items
 - Like item calculation method: Attribute similarity
 - Source model for attribute similarity: Re-estimate attribute weights with New Model
5. Submit the new run from the UI. It suffices for the purpose of TIR if the run only completes the estimation portion successfully and the relevant attribute similarity tables get populated. The execution and results of the base demand and forecast generation steps do not influence the attribute similarity calculations.
6. Once the data is validated in the attribute similarity tables, approve the estimation run by clicking on the 'Approve Demand Parameters' button on the UI. There is no need to approve the base demand and forecast for the purpose of TIR. Only one estimation run remains approved under a run type. When a new estimation run is approved, the previous approved estimation run gets unapproved automatically. TIR uses the attribute similarity data from the approved estimation run under the TIR run type.

Table 23-9 Relevant Queries for Data Validation

```
--query 15: gives merchandise hierarchy type ID for the TIR run type
select * from RSE_FCST_DFLT_PARAMETER where FCST_RUN_TYPE_ID = <<ID of the TIR run
type>> and PARAMETER_NAME = 'PROD_HIER_TYPE';

--query 16: gives location hierarchy type ID for the TIR run type
select * from RSE_FCST_DFLT_PARAMETER where FCST_RUN_TYPE_ID = <<ID of the TIR run
type>> and PARAMETER_NAME = 'LOC_HIER_TYPE';
```

Table 23-9 (Cont.) Relevant Queries for Data Validation

--query 17: gives the PROD_ATTR_GRP_ID that has been copied over from RSE_PROD_ATTR_GRP to the TIR run type based on BUSINESS_OBJECT_ATTR_MD_ID (BOAM_ID), CM_GRP_ID (this is PROD_HIER_ID at the level of CMGRP_LEVEL_ID in RSE_CONFIG), and RSE_PROD_ATTR_GRP.FUNC_ATTR_FLG = 'N'

select * from RSE_FCST_RUN_TYPE_ATTR_XREF where FCST_RUN_TYPE_ID = <<ID of the TIR run type>> and BOAM_ID = <<ID from query 3>> and CM_GRP_ID = <<PROD_HIER_ID from query 4>>;

--query 18: gives the list of estimation runs that are suitable for attribute-similarity based TIR usage (the run of interest should be in this list)

select * from PMO_RUN_HDR where FCST_RUN_TYPE_ID = <<ID of the TIR run type>> and ATTR_WGT_FLG = 'Y' and RUN_STATUS_ID = (select ID from PMO_RUN_STATUS where NAME = 'COMPLETE') order by ID desc;

--query 19: gives the PROD_ATTR_GRP_ID that has been copied over from the TIR run type to the estimation run based on BUSINESS_OBJECT_ATTR_MD_ID (BOAM_ID) and CM_GRP_ID (this is PROD_HIER_ID at the level of CMGRP_LEVEL_ID in RSE_CONFIG)

select * from PMO_RUN_ATTR_XREF where RUN_HDR_ID = <<ID of the estimation run>> and BOAM_ID = <<ID from query 3>> and CM_GRP_ID = <<PROD_HIER_ID from query 4>>;

--query 20: gives the SIM_ID/SRC_ID based on the estimation run and CM_GRP_ID (note that the TRADE_AREA_HIER_ID in this table is the Company/top-most level of the Local Hierarchy)

select * from PMO_SIM where RUN_HDR_ID = <<ID of the estimation run>> and CM_GRP_ID = <<PROD_HIER_ID from query 4>> and ACTIVE_FLG = 'Y';

--query 21: gives the SIM_INTERVAL_ID based on the groupings of the historical weeks (INTERVAL_LENGTH comes from SIM_CALC_INT_LENGTH in RSE_CONFIG)

select * from PMO_SIM_INTERVAL order by START_FISCAL_WK_ID;

--query 22: gives the leaf-level LOC_HIER_ID (Stores) based on the estimation run and SIM_ID

select * from PMO_SIM_LOC_CUSTSEG where RUN_HDR_ID = <<ID of the estimation run>> and SIM_ID = <<ID from query 20>>;

--query 23: gives the leaf-level PROD_HIER_ID (SKUs) and LOC_HIER_ID (Stores) pairs based on the estimation run, SIM_ID, and having the same PARENT_PROD_HIER_ID at the level of CMGRP_LEVEL_ID

select * from PMO_SIM_PR_LOC where RUN_HDR_ID = <<ID of the estimation run>> and SIM_ID = <<ID from query 20>> and PROD_HIER_ID in (select CHILD_PROD_HIER_ID from RSE_PROD_HIER_TC

where HIER_TYPE_ID = <<PARAMETER_VALUE from query 15>> and PARENT_HIER_LEVEL_ID = <<CMGRP_LEVEL_ID from RSE_CONFIG>> and PARENT_PROD_HIER_ID = <<PROD_HIER_ID from query 4>> and CURRENT_FLG = 'Y');

--query 24: gives the leaf-level PROD_HIER_ID (SKUs) and LOC_HIER_ID (Stores) pairs based on the estimation run, SIM_ID, SIM_INTERVAL_ID, and having the same PARENT_PROD_HIER_ID at the level of CMGRP_LEVEL_ID (SLS_IDX is Baseline Sales for the SKU-Store-Interval / Total Baseline Sales for the Store-Interval; SLS_SHARE is Baseline Sales for the SKU-Store-Interval / Average Baseline Sales for the Store-Interval)

select * from PMO_SIM_SLS where RUN_HDR_ID = <<ID of the estimation run>> and SIM_ID = <<ID from query 20>> and SIM_INTERVAL_ID = <<ID from query 21>> and PROD_HIER_ID in (select CHILD_PROD_HIER_ID from RSE_PROD_HIER_TC

where HIER_TYPE_ID = <<PARAMETER_VALUE from query 15>> and PARENT_HIER_LEVEL_ID = <<CMGRP_LEVEL_ID from RSE_CONFIG>> and PARENT_PROD_HIER_ID = <<PROD_HIER_ID from query 4>> and CURRENT_FLG = 'Y');

--query 25: gives the ATTR_WGT based on SIM_ID, BUSINESS_OBJECT_ATTR_MD_ID (BOAM_ID), and PROD_ATTR_GRP_ID

select * from PMO_SIM_ATTR_WGT where SIM_ID = <<ID from query 20>> and BOAM_ID = <<ID from query 3>> and PROD_ATTR_GRP_ID = <<ID from query 4>>;

Table 23-9 (Cont.) Relevant Queries for Data Validation

```
--query 26: gives the ID of the source for attribute similarity
select * from PMO_PROD_SIM_SRC where SRC_NAME = 'PMO';
```

```
--query 27: gives the SIM_VALUE for pairs of leaf-level PROD_HIER_ID (SKUs) based on
PROD_SIM_SRC_ID, HIER_TYPE_ID, SRC_ID, and having the same PARENT_PROD_HIER_ID
(CM_GRP_ID)
select * from PMO_PROD_SIM where PROD_SIM_SRC_ID = <<ID from query 26>> and
HIER_TYPE_ID = <<PARAMETER_VALUE from query 15>> and SRC_ID = <<ID from query 20>> and
CM_GRP_ID = <<PROD_HIER_ID from query 4>> and SIM_VALUE > 0;
```

Fetching Similar Items Based on Attribute Similarity Calculated in AIF

Given the list of items in a shopping cart at a particular store, the TIR for Up-sell service fetches the most similar items from the same store for each item in the cart based on the attribute similarity results.

Table 23-10 Relevant Queries for Data Validation

```
--query 28: gives the leaf-level LOC_HIER_ID for the input Store
select * from RSE_LOC_SRC_XREF WHERE APP_SOURCE_ID = (select ID from
RSE_APP_SOURCE where NAME = 'RA') and HIER_TYPE_ID = <<PARAMETER_VALUE from query
16>> and CURRENT_FLG = 'Y' and LEAF_NODE_FLG = 'Y' and LOC_EXT_KEY = <<input Store>>;
```

```
--query 29: gives the leaf-level PROD_HIER_ID for the input SKU
select * from RSE_PROD_SRC_XREF WHERE APP_SOURCE_ID = (select ID from
RSE_APP_SOURCE where NAME = 'RA') and HIER_TYPE_ID = <<PARAMETER_VALUE from query
15>> and CURRENT_FLG = 'Y' and LEAF_NODE_FLG = 'Y' and LEVEL_ID = <<input SKU>>;
```

Table 23-10 (Cont.) Relevant Queries for Data Validation

```
--query 30: gives the similar items ranked in descending order of similarity
with SIMS as (
SELECT PS.PROD_HIER_ID1 AS DRIVING_PROD_HIER_ID, PS.PROD_HIER_ID2 AS
SIM_PROD_HIER_ID, PS.SIM_VALUE
from PMO_PROD_SIM PS
join PMO_SIM_PR_LOC SPL
on ( PS.PROD_HIER_ID2 = SPL.PROD_HIER_ID )
where SPL.LOC_HIER_ID = <<LOC_HIER_ID from query 28>>
and PS.PROD_HIER_ID1 = <<PROD_HIER_ID from query 29>>
and SPL.RUN_HDR_ID = <<ID of the approved estimation run>>
and SPL.SIM_ID in ( select ID from PMO_SIM S where S.RUN_HDR_ID = <<ID of the approved
estimation run>> and S.ACTIVE_FLG = 'Y' )
and PS.SRC_ID = SPL.SIM_ID
and PS.PROD_SIM_SRC_ID = <<ID from query 26>>
and PS.SIM_VALUE > 0
union all
select PS.PROD_HIER_ID2 as DRIVING_PROD_HIER_ID, PS.PROD_HIER_ID1 as
SIM_PROD_HIER_ID, PS.SIM_VALUE
from PMO_PROD_SIM PS
join PMO_SIM_PR_LOC SPL
on ( PS.PROD_HIER_ID1 = SPL.PROD_HIER_ID )
where SPL.LOC_HIER_ID = <<LOC_HIER_ID from query 28>>
and PS.PROD_HIER_ID2 = <<PROD_HIER_ID from query 29>>
and SPL.RUN_HDR_ID = <<ID of the approved estimation run>>
and SPL.SIM_ID in ( select ID from PMO_SIM S where S.RUN_HDR_ID = <<ID of the approved
estimation run>> and S.ACTIVE_FLG = 'Y' )
and PS.SRC_ID = SPL.SIM_ID
and PS.PROD_SIM_SRC_ID = <<ID from query 26>>
and PS.SIM_VALUE > 0
)
, RANKED_SIM as (
select SIM.*, dense_rank() over (partition by SIM.DRIVING_PROD_HIER_ID order by SIM.SIM_VALUE
desc, SIM.SIM_PROD_HIER_ID) as RNK
from SIMS SIM
)
select PSX.LEVEL_ID as PROD_LEVEL_ID, PSXS.LEVEL_ID as SIM_PROD_LEVEL_ID
from RANKED_SIM RS
join RSE_PROD_SRC_XREF PSX
on ( PSX.HIER_TYPE_ID = <<PARAMETER_VALUE from query 15>>
and PSX.APP_SOURCE_ID = (select ID from RSE_APP_SOURCE where NAME = 'RA')
and PSX.PROD_HIER_ID = RS.DRIVING_PROD_HIER_ID
and PSX.CURRENT_FLG = 'Y'
and PSX.LEAF_NODE_FLG = 'Y')
join RSE_PROD_SRC_XREF PSXS
on ( PSXS.HIER_TYPE_ID = <<PARAMETER_VALUE from query 15>>
and PSXS.APP_SOURCE_ID = (select ID from RSE_APP_SOURCE where NAME = 'RA')
and PSXS.CURRENT_FLG = 'Y'
```

Table 23-10 (Cont.) Relevant Queries for Data Validation

and PSXS.LEAF_NODE_FLG = 'Y'
and PSXS.PROD_HIER_ID = RS.SIM_PROD_HIER_ID)
where RS.RNK <= <<Number of similar items to fetch per item in the shopping cart>>;

24

FAQs

This chapter contains frequently asked questions and their answers.

General Questions

Here are some answers to typical questions of a general nature.

What delimiters are used in the data files?

The inbound data files must be pipe-delimited, while the outbound files are comma-delimited. The delimiters are not configurable.

When are W_RTL_RECLASS_DP_GP_TMP and W_RTL_RECLASS_IT_SC_CL_TMP interfaces used?

These are generally used for implementations that also use RI's data warehouse reporting.

If CREATED_BY_ID and CHANGED_BY_ID columns in every table are set to -1, will it have negative impacts on the functionality?

The value of -1 is the normally expected/used value. The value used will not affect functionality.

CREATED_ON_DT and CHANGED_ON_DT and SRC_EFF_FROM_DT are given values for every table, but since the value is not used functionally, can it be set to a value of NOW?

The only interface where this approach is not appropriate is W_RTL_PROMO_DS. For this interface, the SRC_EFF dates must indicate when the promotion was in effect. (If sales transaction data is not provided, W_RTL_PROMO_DS.dat interface becomes pointless, and therefore defaulting to NOW will be fine everywhere else.)

DT Questions

Here are some answers to typical questions related to demand transference.

Demand transference can make use of similarities derived from customer-linked transactions data. What is the advantage of using this type of similarity, instead of attribute-based similarities?

The benefit to using customer-linked transactions is that the similarities are not dependent on good construction of attributes and attribute values; in addition, the similarities will agree with the CDT if the customer has also implemented CDT. However, keep in mind that to handle the case of new items, attributes are required anyway, so even when using transaction-based similarities, it is still necessary to construct attributes.

What are the disadvantages of using similarities derived from customer-linked transactions data?

Two key disadvantages are:

Retailers want the similarities to be intuitive to their business users, and attribute-based similarities are much better for this because they are more transparent, particularly when the attributes provided by the business users themselves are used.

The requirement for frequent repeat purchases is a high requirement for a category to meet. Many grocery and drug store categories can meet this, but even at grocery or drug stores, there are categories, such as small electronics, that cannot meet this requirement. There are also categories such as batteries or toothbrushes that do have repeat purchases, but the length between purchases is quite long. For such categories, it is probably better to use attribute-based similarities.

Is it possible to use transactions-based similarities for categories where repeat purchases occur but are very infrequent?

It is recommended that you use attributes for such categories, but if there is sufficient data for the category, it is still possible to use transactions-based similarities. You must have either (or both):

- A longer period of history for the category (for example, three years instead of one year) so that there is more chance of catching repeat purchases. Note that this does not necessarily mean that the actual data volume is higher. For example, for batteries, the data volume for three years might be equal to the data volume for six months of dairy, since battery sales have a much lower rate than dairy.
- A large number of customers (at least 100,000) for each segment of the category, if you are implementing segments, or just a large number of customers (at least 100,000) for the entire category, if you are not implementing segments. Having such a large number of customers increases the probability that a significant fraction of them will have made repeat purchases in the category in your historical data, even if each customer did not make many repeat purchases. For example, you could have 100,000 customers, each of whom made only one repeat purchase.

How can grouped attributes be used in the AIF platform?

Grouped attributes are recommended only for CDT, not DT. For CDT, the guideline is that attributes must have no more than seven values. Grouping is a way to achieve this goal, as it groups together many attribute values into a single value. Having attributes with a large number of attribute values can result in very large CDTs, especially if the attribute occurs at the top of the tree. This general guideline of seven values is a way to keep CDTs to a reasonable size. Note that retailers seem to prefer having large CDTs with ungrouped attributes.

What attributes are typically good candidates for grouping?

The most common attributes are Brand, Flavor, and Color, because these attributes typically have a large number (that is, more than 50) of values. Keep in mind, that retailers typically do not want grouped attributes.

How can the Brand attribute be grouped?

Brands can, for example, be grouped into High End, Mainstream, and Budget. The grouping must be driven by how the retailer thinks of the category and thinks of brands in the category.

Does the software automatically determine which attributes are functional fit? Can the software help me determine which attributes are functional fit ones?

There is no way to automatically detect functional-fit attributes, or close-to-functional fit attributes. The determination is done today using business users' intuition, not science.

Are there other uses of functional-fit attributes besides the obvious examples like size in wiper blades and clothing? What about functional-fit attributes for food items?

Functional-fit attributes are useful for items such as food for expressing strong customer preferences. For example, caffeinated can be set as a functional-fit attribute for the Coffee category to eliminate transference between caffeinated and de-caffeinated coffee. While it is possible that a small minority of people might transfer purchases of one to the other, it is unlikely enough that, when planning an assortment, transference should simply be eliminated. Note that this is based on the business user's intuition, since there is no automatic way to detect these situations.

What are the special considerations for categories containing very few items, such as 20 or fewer?

Such categories are very rare; typically, categories have hundreds to thousands of items. Small categories may cause a problem in determining demand transference because their historical data may contain very few assortment changes. The more items a category has, the more likely assortment changes will have occurred; conversely, small categories may have very few historical assortment changes, which makes determining transference in the category difficult. One solution is to merge the small category into a bigger category that has similar items. For example, merge a specialty bacon category containing 20 items with the regular bacon category (since both are bacon). This amounts to applying the assortment elasticity of the larger bacon category to the specialty bacon, and if the specialty bacon by itself has very few assortment changes, this is a reasonable solution.

How is new-item introduction handled by demand transference? Which cases of new items are handled?

The case the application handles is introducing new items at an existing store for an existing category that already has attributes. For this case, all that is required for the new item are its attribute values. No sales history is required for the new item. From the attribute values assigned to the new item, the similarity of the new item to existing items is calculated. This similarity is used to forecast a base rate of sale for the new item based on the sales rates of existing similar items. This base rate of sale is then used in demand-transference calculations.

How is W_RTL_PRODUCT_BRAND_DS interface used by DT?

Brand can be provided as an ITEMUDA, or it can be provided via W_RTL_ITEM_GRP1_DS with a PROD_GRP_TYPE=BRAND. If it is done as a PROD_GRP_TYPE=BRAND, then this interface receives the master list of Brands, which the W_RTL_ITEM_GRP1_DS refers to. If Brand is just provided as an ITEMUDA, then there is no need for this file.

How is W_RTL_PRODUCT_COLOR_DS interface used by DT?

In the same way as the W_RTL_PRODUCT_BRAND_DS interface, only the PROD_GRP_TYPE is COLOR in this case.

Attribute Processing Questions

Here are some answers to typical questions related to attribute processing.

Is it allowed for RSE_PROD_ATTR_VALUE_XREF_STG to have attribute values which is not linked to items in W_RTL_ITEM_GRP1_DS? For example, given that in a certain season W_RTL_ITEM_GRP1_DS has following item/attributes: Item A Attribute 1, Item B Attribute 2, Item C Attribute 3, and Item D Attribute 4. RSE_PROD_ATTR_VALUE_XREF_STG should have Attribute 1,2,3,4. In the next season,

Item C is dropped: Item A Attribute 1, Item B Attribute 2, and Item D Attribute 4. In this case, RSE_PROD_ATTR_VALUE_XREF_STG should be rebuilt to have only Attribute 1,2,4?

Yes, it is allowed. The load will not fail if there is no data found, so it is not a data validation rule that requires there to be a matching attribute in W_RTL_ITEM_GRP1_DS. If there are no products associated with the attribute value, then the attribute value will not be used.

Can you have records with NULL for all the following five "VALUE" columns in RSE_PROD_ATTR_VALUE_XREF_STG? So, records will have value only for PROD_ATTR_VALUE_KEY and ATTR_VALUE_EXT_CODE columns: MIN_ATTR_NUM_VALUE, MAX_ATTR_NUM_VALUE, ATTR_STRING_VALUE, MIN_ATTR_DATE_VALUE, and MAX_ATTR_DATE_VALUE.

The interface provides multiple choices for identifying the attribute value. You must pick the one approach that works for the specific attribute and the remaining values are NULL. So for attributes that relate to data in W_RTL_ITEM_GRP1_DS, you provide a value for ATTR_VALUE_EXT_CODE and leave the other five indicated Attribute Value columns as NULL.

Lifecycle Pricing Optimization Questions

Here are some answers to typical questions that you may encounter.

What are the recommended steps for preparing the environment for LPO?

Follow the *Oracle Retail Analytics and Planning Implementation Guide* and *Oracle Retail AI Foundation Implementation Guide* for setup, as the applications are designed to work together. For LPO specifically, the Lifecycle Pricing Optimization chapter in the *AI Foundation Implementation Guide* covers the detailed configuration steps.

Why does the initial setup and historical load take so long to process?

Initial data loads, especially on large datasets such as inventory, sales, or transaction history, can be time consuming due to data volume and necessary validations. Several hours of multi-month data is not uncommon particularly if you are loading millions of records. To ensure better performance:

- Ensure data files are clean and formatted per specifications before uploading them.
- Where possible, chunk files or perform incremental uploads.
- Take into account your compute, storage, and data volumes.

After the setup and configuration, what steps should be taken to confirm that LPO and POM are ready for use?

For LPO:

- Confirm all prerequisite data interfaces are loaded and validated.
- Verify roles and permissions.
- Perform an initial scenario / test to confirm data is flowing and outputs are generated.
- More details can be found in the Lifecycle Pricing Optimization chapter in this guide.

For Process Orchestration and Monitoring (POM) availability, see the Process Orchestration and Monitoring documentation on Oracle Help Center.

How can the environment be set up to support multiple languages across different countries and browsers?

This is the same across all Oracle Retail Cloud applications and Globalization / Localization requirements. The browser language is the driver. You should be able to test this on your computer by switching the language in the browser.

What is the recommended approach to configure a forecast run for LPO?

In the *Oracle Retail AI Foundation User Guide*, see the Forecasting Methods in AIF chapter. It goes through the different forecasting methods. Lifecycle Method is typically the forecast we would recommend that will produce reliable forecasts at the lowest levels of the Merchandise and Location hierarchies, accounting for prices and markdowns but with a larger breadth of data needs.

Where can I find more information on Business Rules for LPO?

See the appendixes in the *Oracle Retail Lifecycle Pricing Optimization User Guide*.

How should business rules be configured, and what is the impact of using or not using forecasts?

The approach to managing forecast-based and non-forecast-based rules depends on how the business intends to use the solution. Non-forecast-based rules apply only to Regular Price.

In a price-zone, there are stores with different inventory levels. Some stores have very high inventory levels, and others have low inventory levels. Does LPO recommend different recommendations for the zone and such stores separately?

This is not possible in the current release. You can configure LPO to run either at the price-zone level or at the store-level. If the price-zone level is selected, then all the stores within that price-zone will receive the same price recommendations. If you want to have store-level recommendations, then you must configure the run-level to be at the store-level and let the algorithm dictate the recommendations. If the stores share very similar customer behavior, then it may be possible that they will receive similar recommendations.

I am not interested in the promotions; can the product handle only markdowns?

Yes, the product can handle only markdowns. You can define all the periods to allow markdowns, and no period should have promotions allowed.

I have a planned promotion and I know when the promotion should occur, but I do not know the items and the depth. Can the product recommend items and the depth?

No, the product requires planned promotion to specify the items, depth, and when the promotion should occur. However, if the retailer wants the product to recommend items and depth, then you can specify the promotion and markdown calendar using the interface provided and specify when the promotions and markdowns can be provided. LPO will determine which items should be promoted or the markdown among the set of periods provided. Note that you can also use the relative calendar option through the UI to specify the calendar.

Should I use Absolute Calendar or Relative Calendar?

The two calendars are provided for flexibility in handling two use cases. When all the items are introduced at almost the same time and exit at almost the same time, then it makes sense to use the absolute calendar. You can also use the relative calendar in such a use case. When the items are introduced at periodic intervals (for example, summer merchandise is introduced

in May, June, and July) then it makes sense to use relative calendar. For example, the absolute calendar in such a scenario would probably say May, June as Regular and July as Clearance. This means that items that were introduced in July are going directly into a Clearance season. However, the relative calendar begins after the item is introduced. It is defined as the percentage of regular season and the percentage of the markdown season, based on the total length of the season available for the item.

Is the Style, Color needed as part of Merchandise Hierarchy?

Since the product supports fashion apparel, it is expected that the style and color be provided as part of merchandise hierarchy. If the retailer carries non-apparel items, then the retailer can specify some generic value for style and color, such as not applicable.

How does LPO handle the online channel?

The retailer can specify the online channel as a separate price zone. This means that the price recommendations for brick-and-mortar stores and online-stores will be different. If the retailer does not wish to have different price recommendations, then it is suggested that you combine the online channel into a brick-and-mortar store.

Can LPO handle complex promotions?

LPO handles only simple promotions (for example, percentage off). It does not handle complex promotions like BOGO.

Can LPO come up with bundles for complex promotions? How can this be handled?

LPO cannot come up with bundles for a promotion. A bundle can be specified using Product Groups under Business Rules.

Is budget constraint over the effective week or over the entire life of the item?

Budget is assumed to be for the entire remaining life of the item. The user can specify either different budgets for Promotions and Markdowns or a combined budget. Note that a planned promotions can be excluded or included from this budget.

What happens when a user accepts or rejects a recommendation? Do the numbers change for that week? Does it optimize the entire life again?

In the current release, the metrics in the Results or Manage Recommendations do not reflect the accept, reject, or override price recommendations. However, when the user selects the Recalculate button, then all the metrics are recalculated taking into account the price modifications done by the user. But, when the user selects Reoptimize, then the price modifications are ignored.

Can LPO handle any type of price-zone?

Yes. In the current release, LPO can accept price-zones that are defined as any combination of locations and merchandise.

How does LPO handle multi-currency support?

LPO accepts the historical sales data in the local currency provided. Further, the location in the application can be specified a country locale. Each application run has a location, which is used to tie the price and price-related metrics (for example, revenue) to the specified currency. However, if you want to configure the run to be at a level that spans multiple currencies, then you must specify one currency.

How does LPO support dynamic clustering?

Any time you change the price-zone or re-cluster the customer segments, the change must be re-loaded as part of new hierarchies to LPO. It is considered a major change to the application as it affects the core data elements. The application will be re-calculating the demand parameters and other relevant optimization inputs, based on the new hierarchies. Further, the UI does not offer support to re-group stores within a price-zone based on certain attributes, for example, group stores by sell-through.

How are product images loaded?

Product images can be loaded via the interface provided. LPO expects that the images will be loaded on an image server, which can be a local server. An URL is provided with the image name that can be accessed by the application. The application searches for the image and populates the UI with the relevant image for that item in the Targeted Offer screen.

A

Appendix: Innovation Workbench Workshop

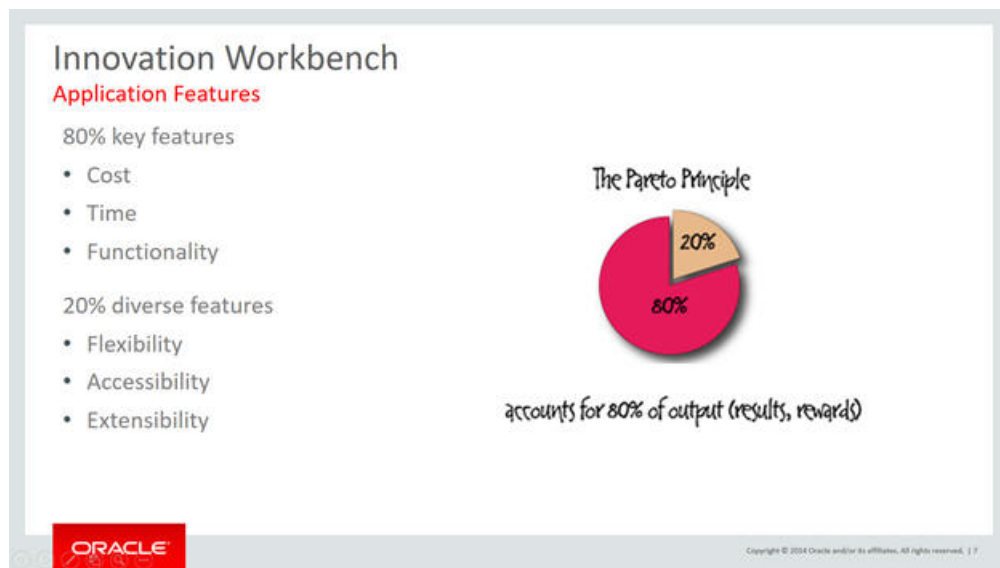
Science Innovation Workbench is a workspace that provides read-only access to the application data objects and clean data by using Oracle APEX. This extension is a workspace for advanced analytics users to add new implementations by using Oracle Advanced Analytic (Oracle R/ODM) algorithms that are implemented as SQL/PLSQL functions.

This appendix provides instructions for using the interface.

Overview

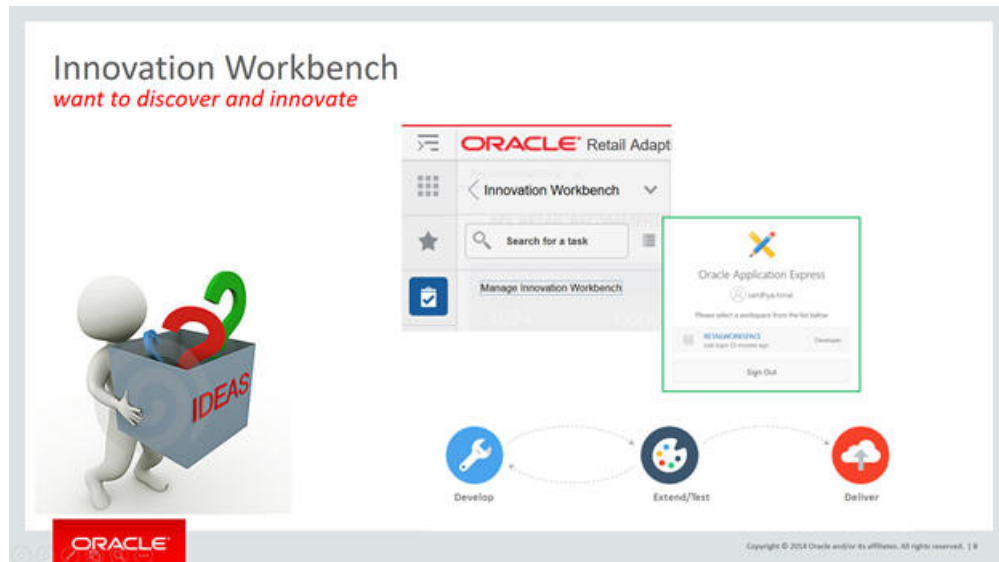
Open Analytics: understanding why, what, and how.

Figure A-1 Overview



Innovation Workbench helps in extending and delivering 20 percent of these features. It provides tools to learn, discover, and innovate existing application features.

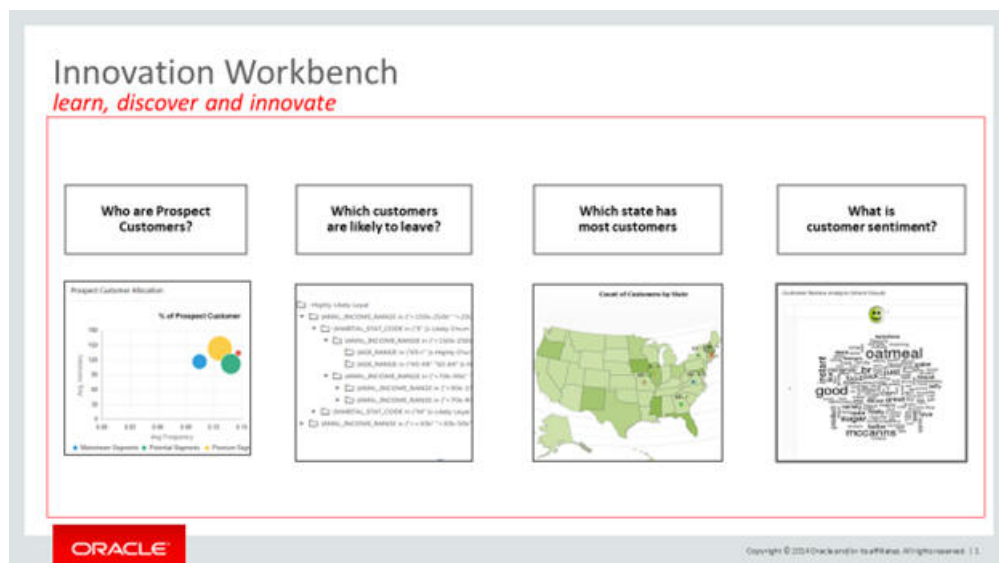
Figure A-2 Discover



Innovation Workbench helps answer questions such as:

- Which customers are likely to leave?
- What is customer feedback?
- Who are the prospect customers?

Figure A-3 Learn

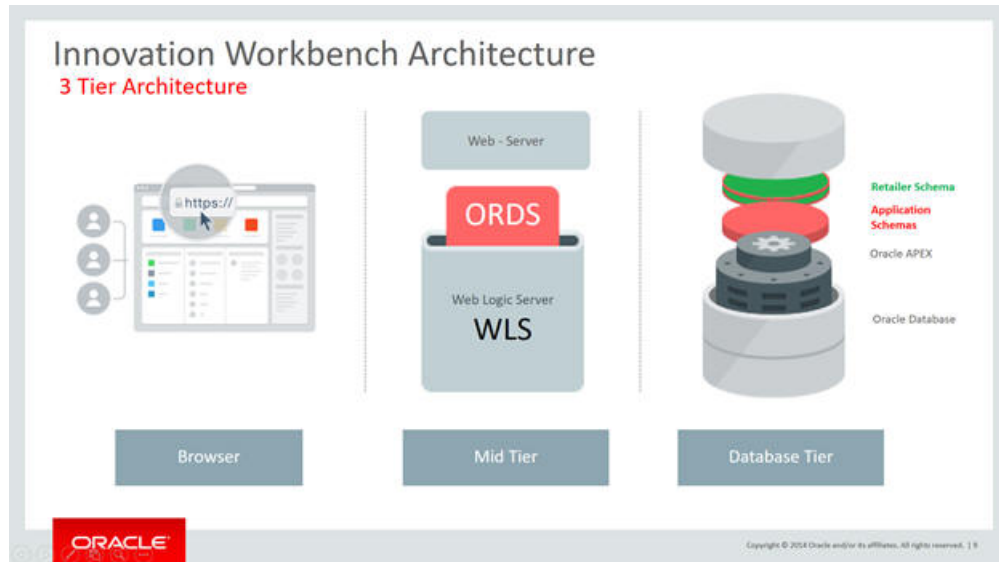


With Innovation Workbench, the following new components are available:

- Retailer schema (see green slice in [Figure A-4](#))
- Retailer workspace

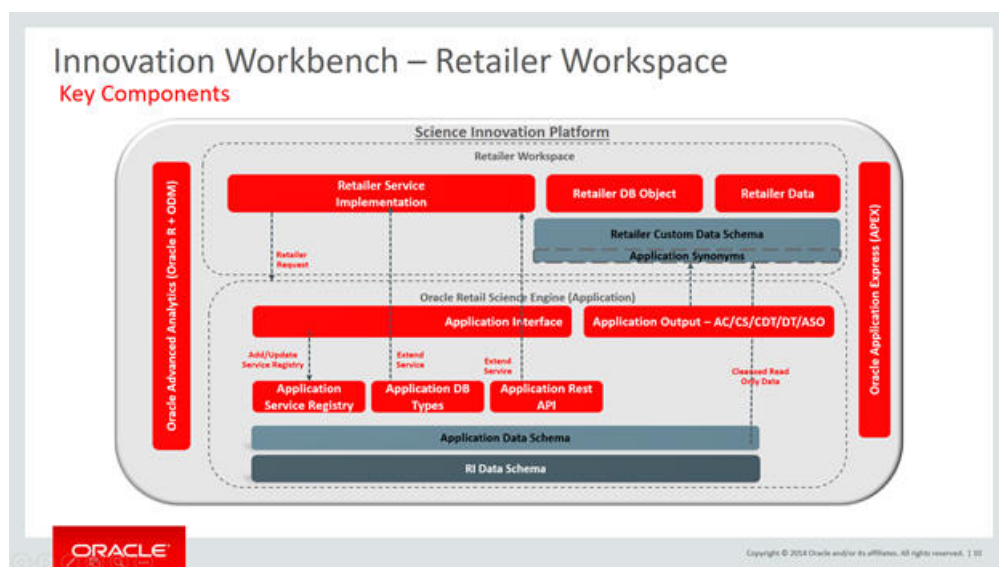
- SQL Workshop
- Application Builder

Figure A-4 Architecture



The Retailer Schema has read-only access to the Application Schema. All permissions are controlled using database permissions.

Figure A-5 Key Components



Roles for the analyst, scientist, developer, and integration are as follows:

- Oracle Developer identifies which objects can be accessed.

- Administrator manages users.
- Retailer Developer uses data shared, extend, explore and mine data.

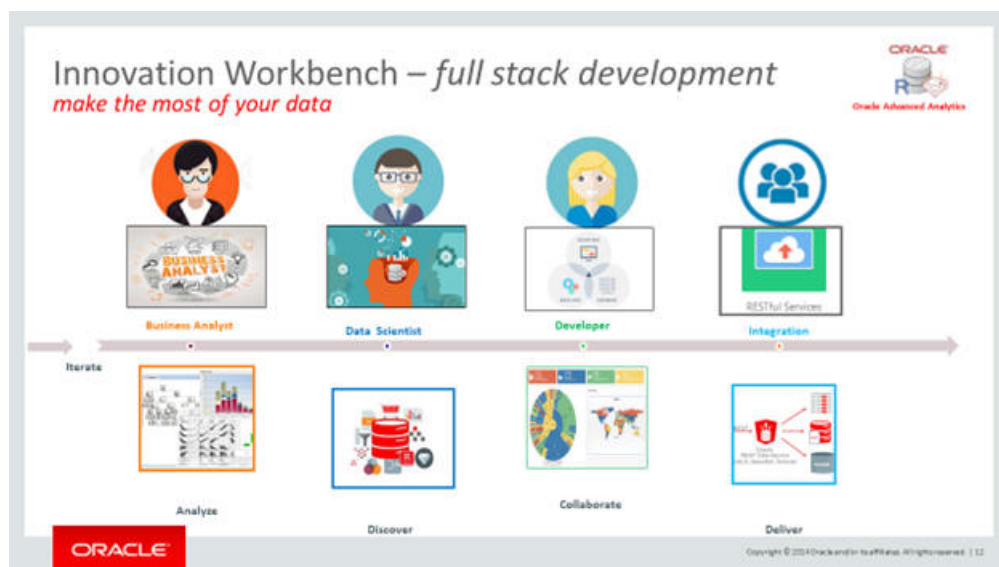
Figure A-6 User Roles



As part of Innovation Workbench, a retailer developer has the following roles:

- Business Analyst, explores data, gains insights
- Data Scientist, discovers patterns in data mining using ODM
- Developer, develops applications and shares insights via UI and web services

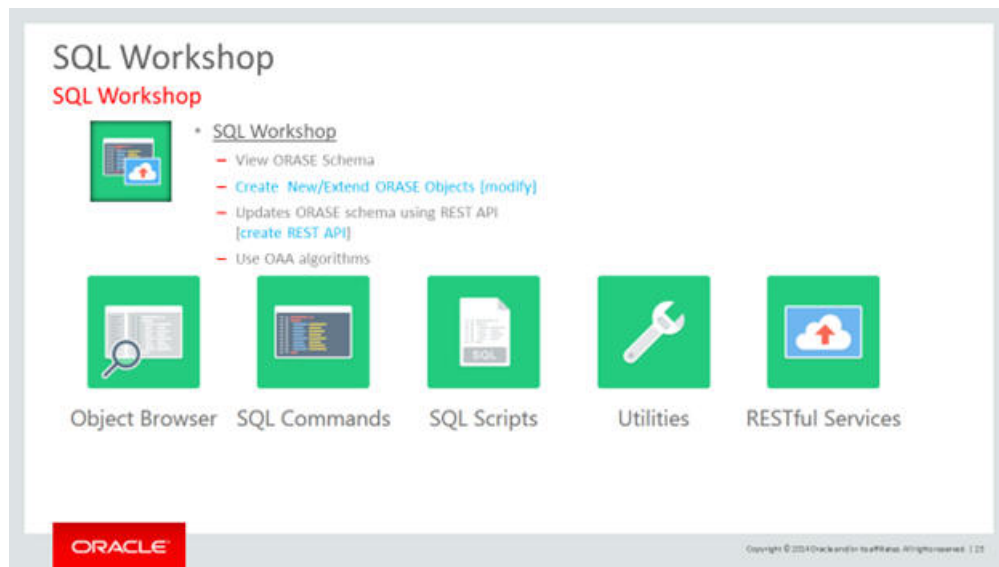
Figure A-7 Full Stack Development



Key components

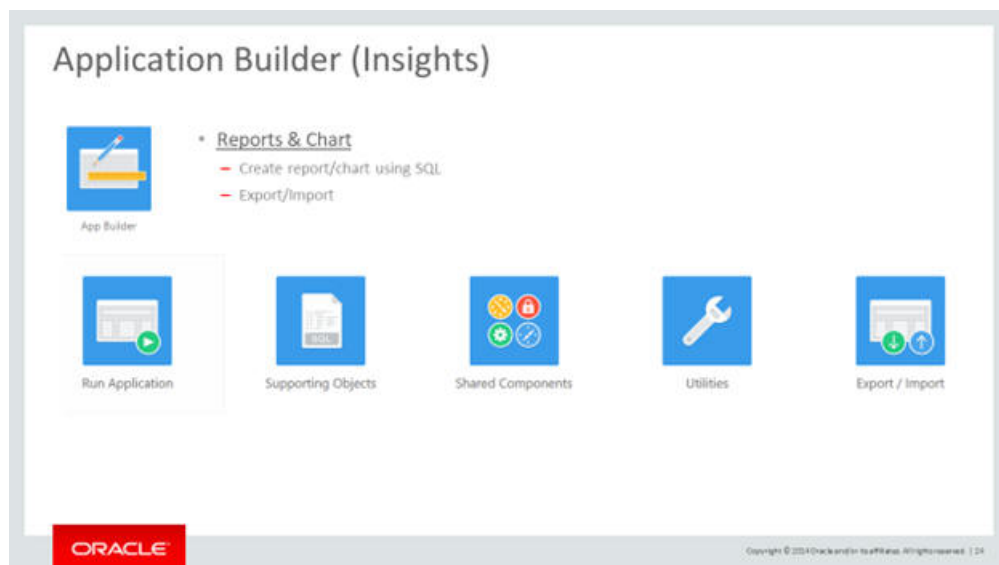
The SQL Workshop provides tools to view and manage database objects.

Figure A-8 SQL Workshop



The Application Builder provides tools to create reports and charts using wizards and page designers.

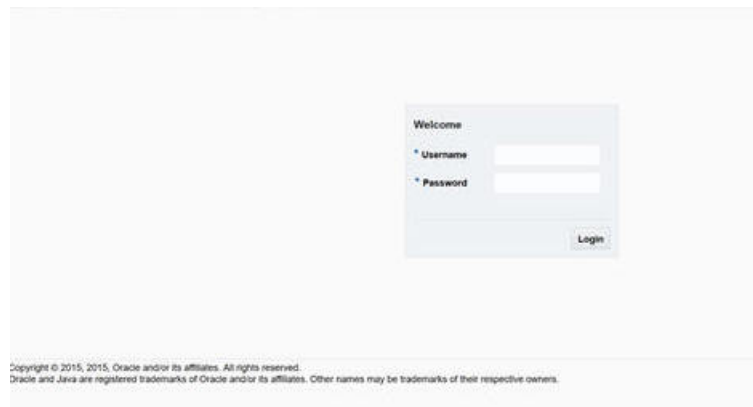
Figure A-9 Application Builder (Insights)



Lab Innovation Workbench

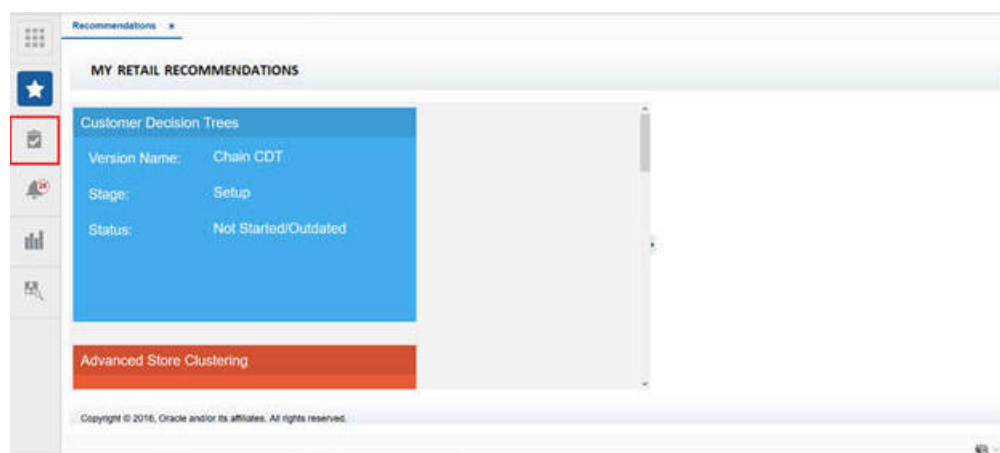
1. Login using `http:// <url>`
Login/Password

Figure A-10 Login



This takes you to the application.

Figure A-11 Application



2. Select Innovation Workbench.

Figure A-12 Select Innovation Workbench



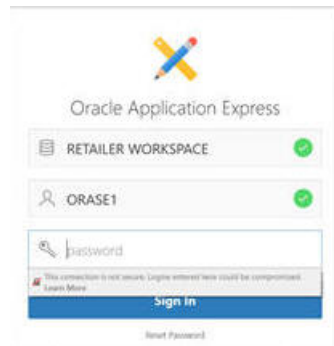
3. Select Manage Innovation Workbench.

Figure A-13 Manage Innovation Workbench



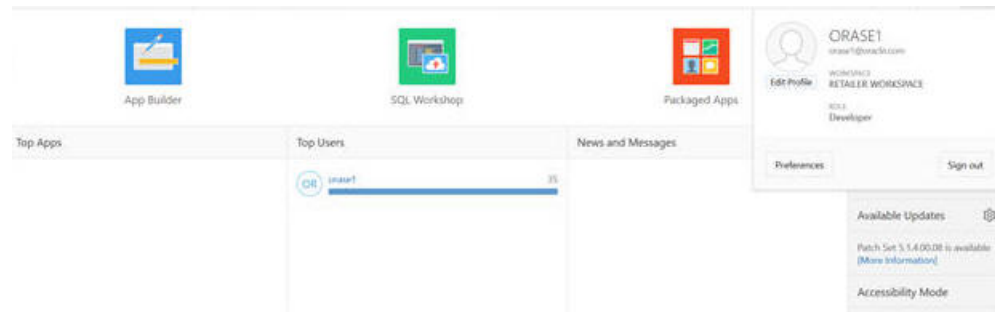
This takes you to Retailer Workspace where you can log in with same login and password credentials.

Figure A-14 Retailer Workspace Login



Retailer Workspace has a dedicated schema and work area to do full stack development.

- Dedicated Schema for retailer
 - Granting and revoking database object privileges
 - Dedicated service plan and consumer group for retailer workspace
 - Access to database utilities and scheduling for long-running transactions
- SQL Workshop
- Application Builder

Figure A-15 Retailer Workspace

Lab Analysis

Browse Existing Data

In this exercise you will browse the Application schema objects using SQL Workshop.

- Access Read Only
- Cleansed
- Aggregated
- Filtered
- Sample
- Inputs and Outputs

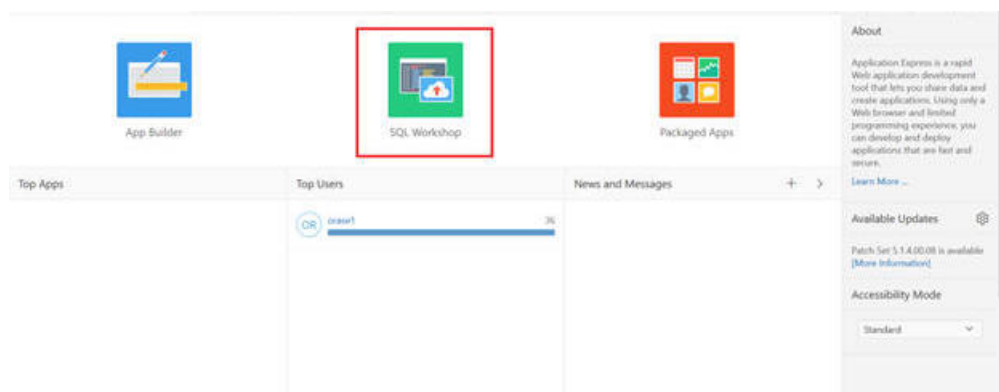
Figure A-16 Application Schema Objects

Figure A-17 SQL Workshop Object Browser

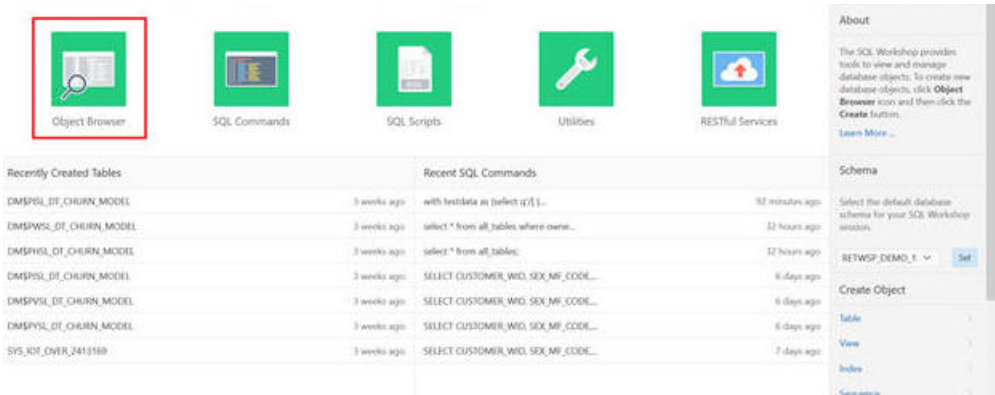
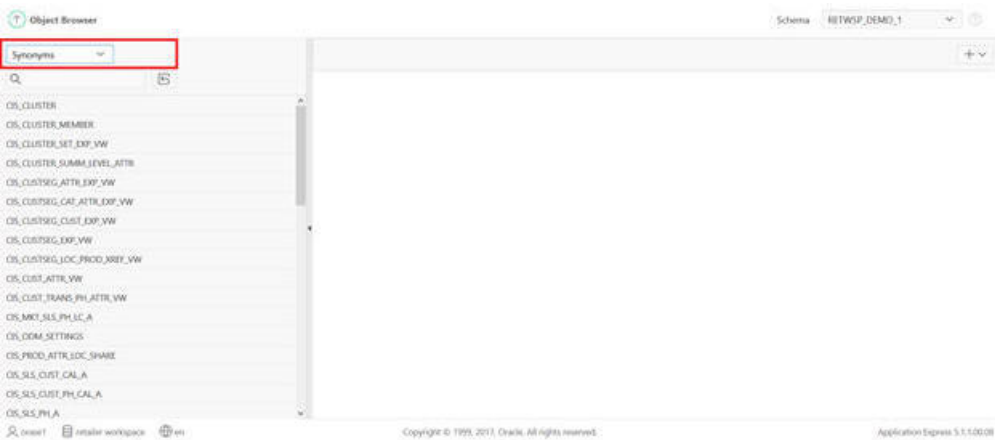


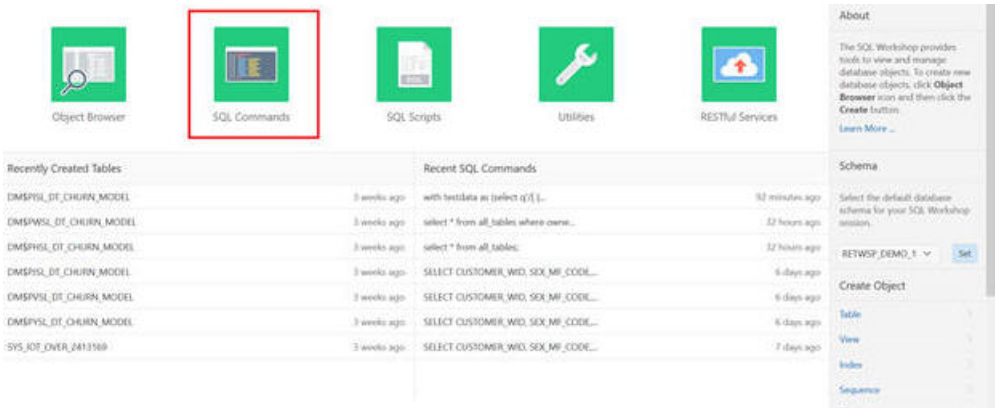
Figure A-18 SQL Workshop Object Browser Synonym



Note that this dedicated schema is only associated with the Application Schema and cannot be extended to any other new schema.

Data

Figure A-19 SQL Workshop SQL Command



This SQL has been saved, so you can type sql in the command and execute and save it.

Figure A-20 SQL Workshop SQL Command Save SQL

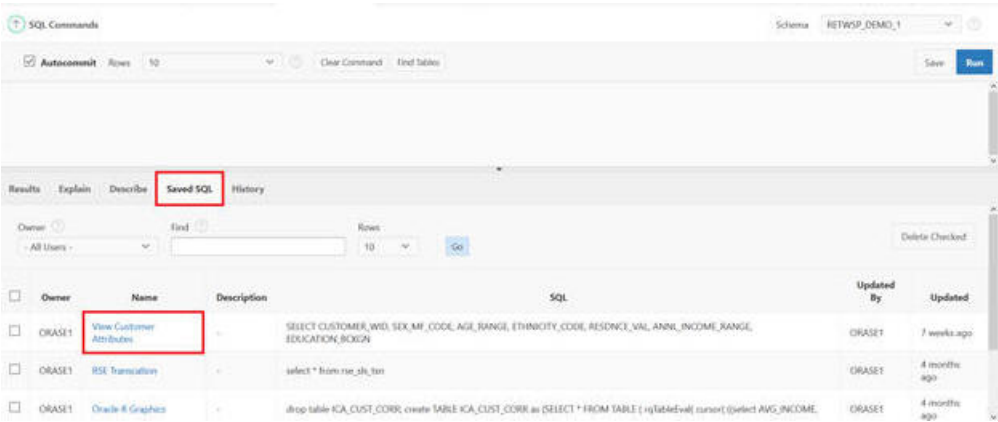


Figure A-21 SQL Workshop SQL Command Saved SQL View Customer Attributes

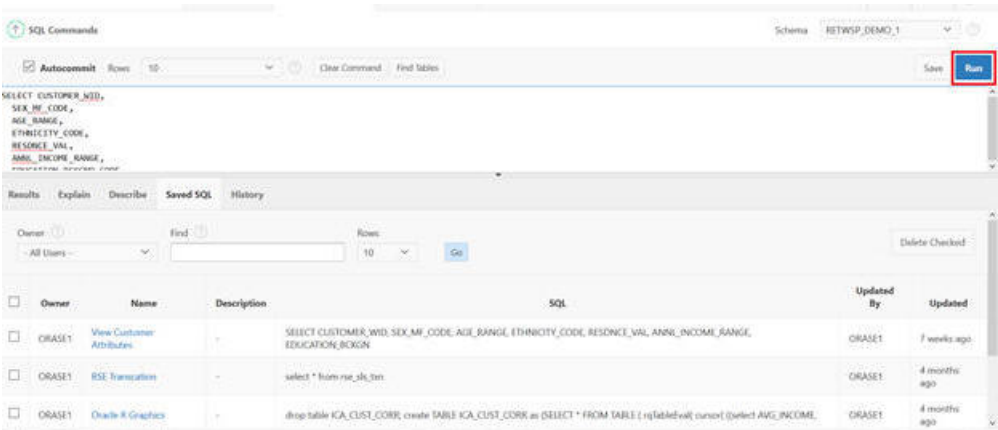


Figure A-22 Browse Data

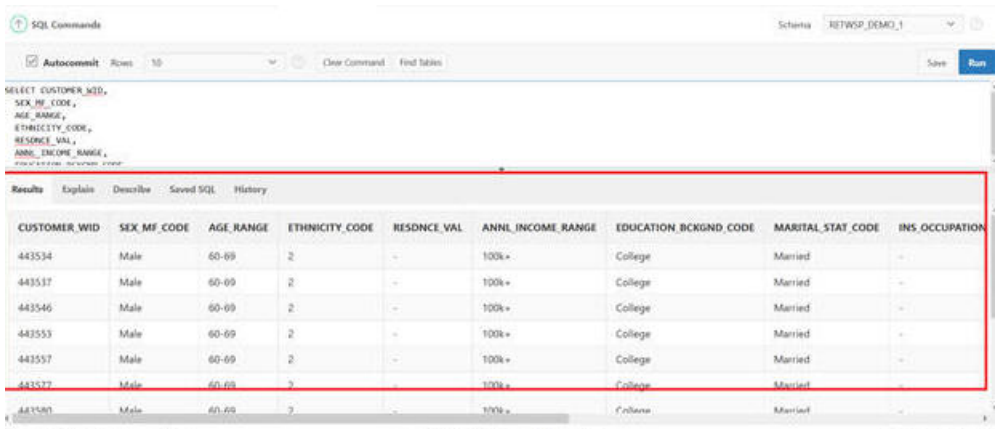
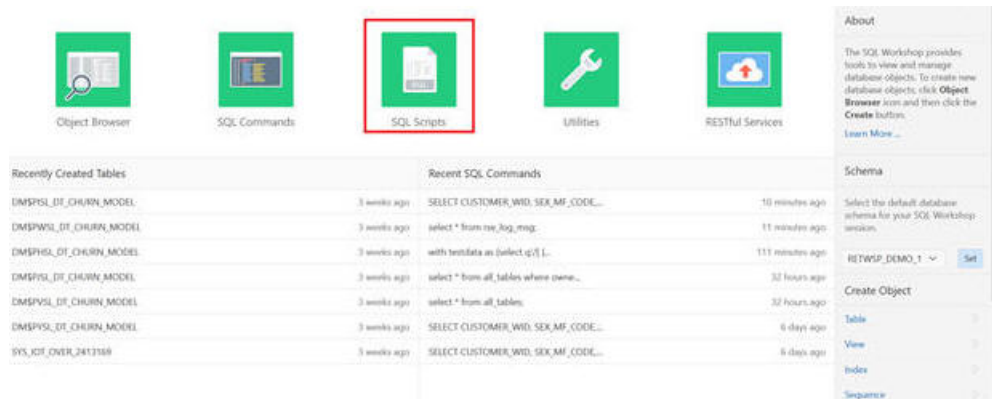


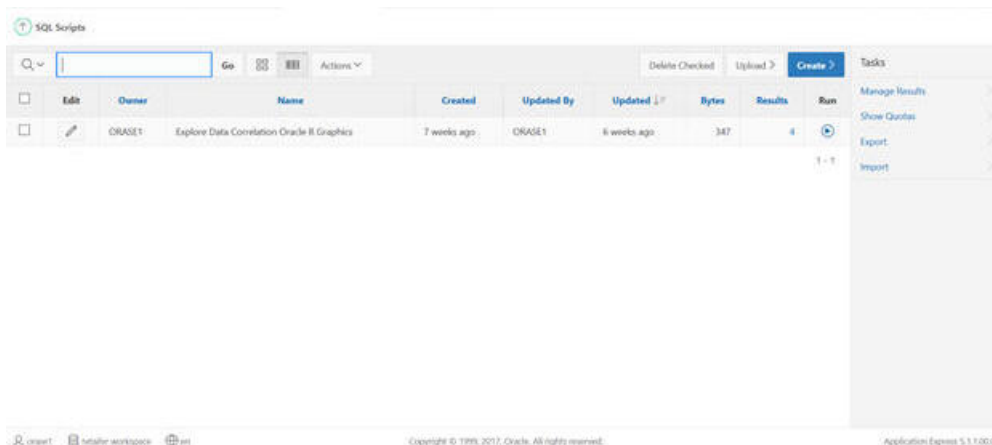
Figure A-23 SQL Workshop SQL Scripts



Using this you can do the following:

- Upload Scripts
- Create Scripts
- Edit Scripts

Figure A-24 Scripts



- Load data
- Use ETL feeds
- Use system feeds wherever possible. If the application feed is not available, then ReST objects can be used.

Using ReST to Load Data

Data can be loaded to the retailer schema using the ReST API in JSON format. These must be batched from the client end while uploading data into application.

SQL Workshop ' SQL Command

Create table that will hold Customer Reviews

```

DROP TABLE retwsp_cust_prod_rev_stg;

CREATE TABLE retwsp_cust_prod_rev_stg(
  json_str clob
);

DROP SEQUENCE retwsp_cust_prod_rev_seq
CREATE SEQUENCE retwsp_cust_prod_rev_seq
  START WITH      1
  INCREMENT BY    1
  NOCACHE
  NOCYCLE;
reviewerID varchar2(255) NOT NULL,

DROP TABLE retwsp_customer_product_review;
CREATE TABLE retwsp_customer_product_review
(

```

- reviewid NUMBER NOT NULL,
- reviewerID varchar2(255) NOT NULL,
- asin varchar2(255) NOT NULL,
- reviewerName varchar2(255) NOT NULL,
- helpful varchar2(255) NOT NULL,
- reviewText varchar2(4000) NOT NULL,
- overall number(2, 0) NOT NULL,
- summary varchar2(255) NOT NULL,
- unixReviewTime NUMBER(10) NOT NULL,
- reviewTime VARCHAR2(255) NOT NULL

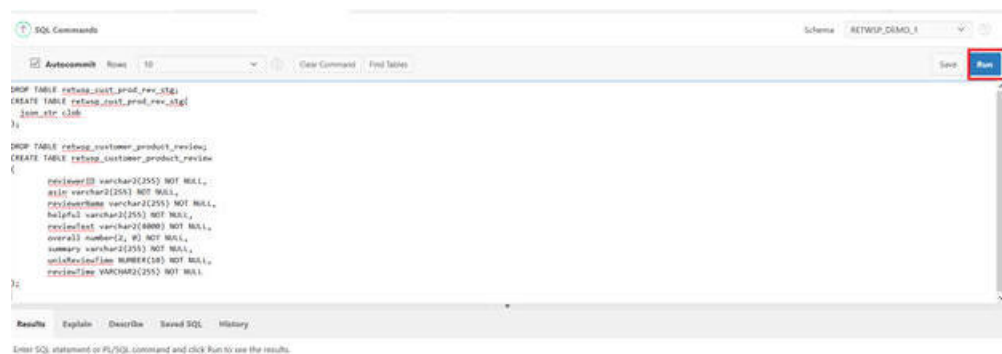
```

);

```

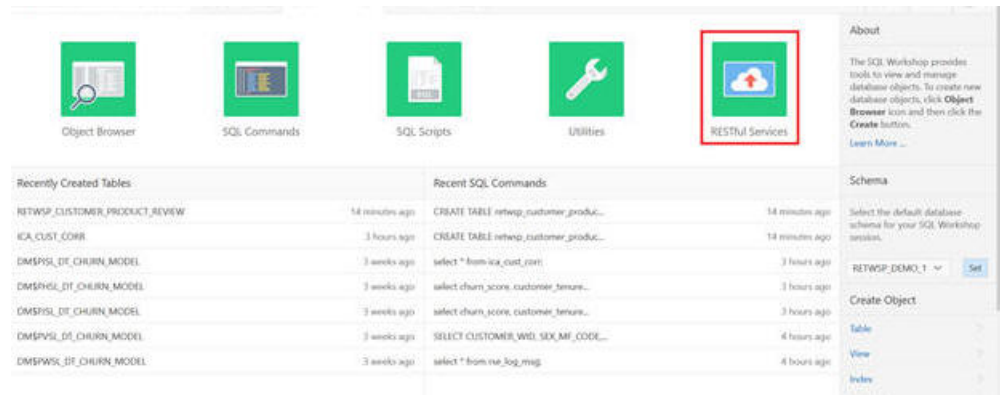
1. Copy and paste the above content and click **Run**.

Figure A-25 Load Data Script



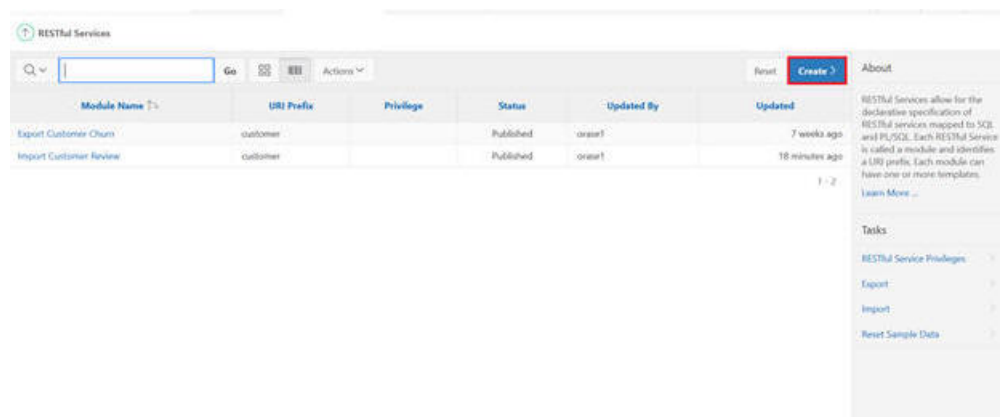
2. Create ReST API, which allows you to load customer reviews in the above table.

Figure A-26 Create REST API



3. Click **Create**.

Figure A-27 Create



4. Here is the code to populate the JSON data to the relational table.

```

Declare
customer_review_lblob blob;
customer_review_cblob clob;
begin
    customer_review_lblob := :body;
    customer_review_cblob := wwv_flow_utilities.blob_to_clob(customer_review_lblob);
    -- insert into stage table
    insert into retwsp_cust_prod_rev_stg values (customer_review_cblob);

    -- move to target table
    insert into retwsp_customer_product_review (reviewid, reviewerID, asin,
reviewerName, helpful, reviewText, overall, summary, unixReviewTime, reviewTime)
with review_data
as (select json_str from retwsp_cust_prod_rev_stg)
select retwsp_cust_prod_rev_seq.nextval, j.*
from review_data rd,
json_table(rd.json_str, '$[*]'
columns (reviewerID varchar2 path '$.reviewerID'
, asin varchar2 path '$.asin'
, reviewerName varchar2 path '$.reviewerName'
, helpful number path '$.helpful'
, reviewText varchar2 path '$.reviewText'
, overall number path '$.overall'
, summary varchar2 path '$.summary'

```

```

        ,unixReviewTime number path '$.unixReviewTime'
        ,reviewTime varchar2 path '$.reviewTime'
    )
) ;
delete from RETWSP_TMP_CUST_PROD_RW;
commit;
end;

```

5. Provide the essential elements, including name, URI prefix, and template.

Figure A-28 ReST Information

6. Select the method. In this case it is POST, since the data is getting updated.

Figure A-29 Select Post

7. Populate the name for the RESTful Services Module. URI Prefix as customer and URI Template/reviews. The / is required.

8. Add resource handler method - POST and source type - PL/SQL

9. Declare


```

customer_review_lblob blob;
customer_review_cblob clob;
begin
    customer_review_lblob := :body;
    customer_review_cblob := wwv_flow_utilities.blob_to_clob(customer_review_lblob);

```

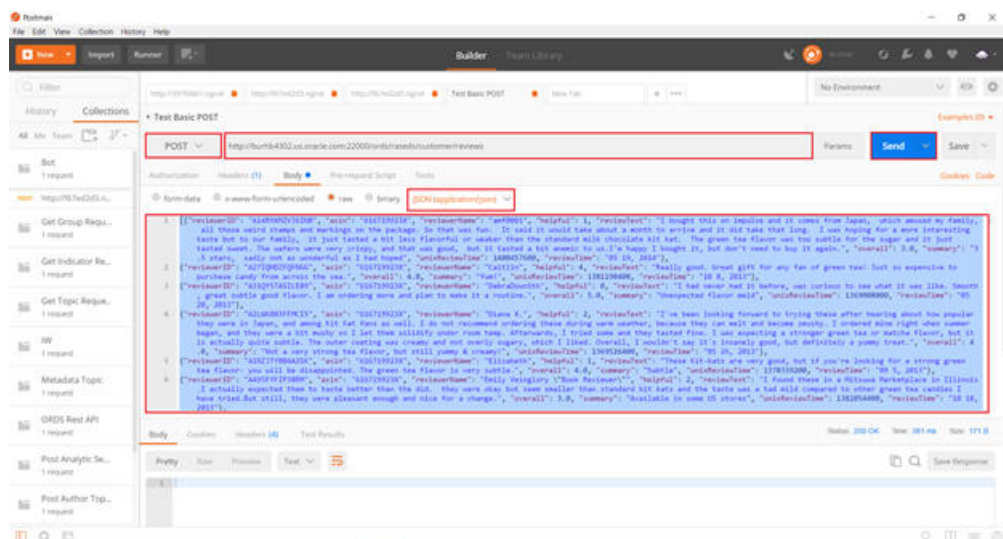
```
-- insert into stage table
insert into retwsp_cust_prod_rev_stg values (customer_review_cblob);

-- move to target table
insert into retwsp_customer_product_review (reviewid, reviewerID, asin,
reviewerName, helpful, reviewText, overall, summary, unixReviewTime, reviewTime)
with review_data
as (select json_str from retwsp_cust_prod_rev_stg)
select retwsp_cust_prod_rev_seq.nextval, j.*
from review_data rd,
json_table(rd.json_str, '$[*]'
columns (reviewerID varchar2 path '$.reviewerID'
, asin varchar2 path '$.asin'
, reviewerName varchar2 path '$.reviewerName'
, helpful number path '$.helpful'
, reviewText varchar2 path '$.reviewText'
, overall number path '$.overall'
, summary varchar2 path '$.summary'
, unixReviewTime number path '$.unixReviewTime'
, reviewTime varchar2 path '$.reviewTime'
)
) j;
delete from retwsp_cust_prod_rev_stg;
commit;
end;
```

10. Post customer reviews using following the data <http://<url>>

This posting can be done using POSTMaN or any ReST Client (such as java application posting data into Retailer Schema) This step requires PostMan or a client that will push reviews. Download and Install PostMan. If it cannot be installed, go to the next step. <https://www.getpostman.com/apps>

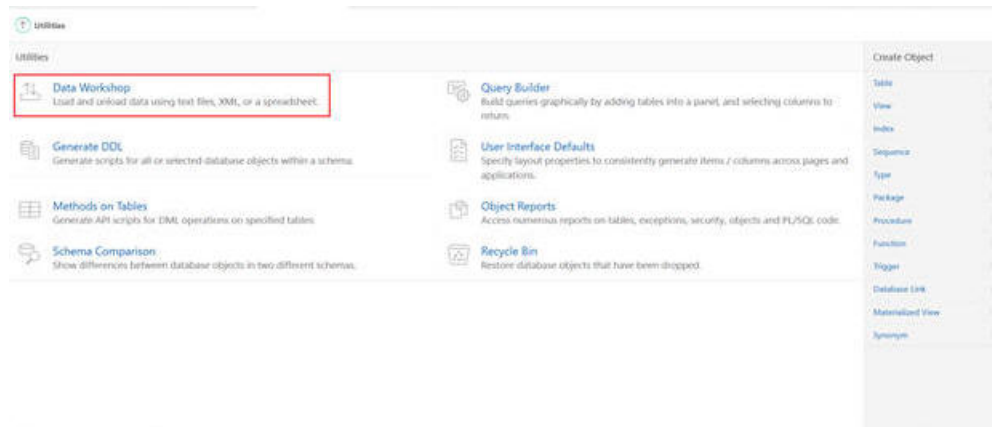
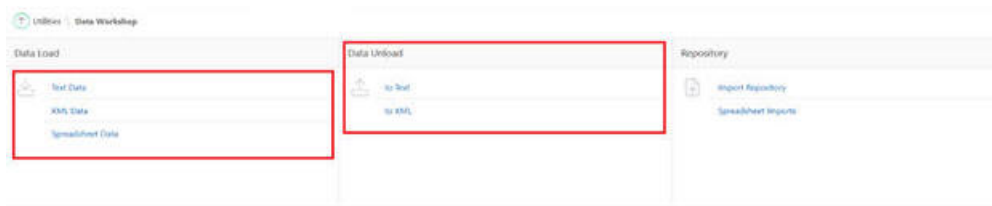
Figure A-30 Post



11. Select * from retwsp_customer_product_review;

Other Methods for Loading Data

These methods must be used judiciously, with smaller datasets.

Figure A-31 Data Workshop**Figure A-32 Load and Unload Data**

Prepare Data

Key components that can be used for refining data are Create Train and Test dataset.

The followings SQL shows how stratified split is used to preserve the categorical target distribution in the resulting training and test dataset.

Key items below are:

- target column. It must be categorical type.
- case id column. It must contain unique numbers that identify the rows.
- input data set. {percent of training dataset} - percent of training dataset. For example, if you want to split 60% of input dataset into a training dataset, use the value 60. The test dataset will contain $100\% - 60\% = 40\%$ of the input dataset. The training and test dataset are mutually exclusive.

Train Datasets

```
create or replace view retwsp_rf_cust_attr_test_vw as (SELECT v1.* FROM (
-- randomly divide members of the population into subgroups based on target classes
SELECT a.*, row_number() OVER (partition by EDUCATION_BCKGND_CODE ORDER BY
ORA_HASH(CUSTOMER_WID)) "_partition_caseid"
FROM
```

```

retwsp_rf_cust_attr_vw a) v1, (
-- get the count of subgroups based on target classes
SELECT
EDUCATION_BCKGND_CODE,
COUNT(*) "_partition_target_cnt" FROM
retwsp_rf_cust_attr_vw GROUP BY EDUCATION_BCKGND_CODE) v2
WHERE v1.EDUCATION_BCKGND_CODE = v2.EDUCATION_BCKGND_CODE
-- random sample subgroups based on target classes in respect to the sample size
AND ORA_HASH(v1."_partition_caseid", v2."_partition_target_cnt"-1, 0) <=
(v2."_partition_target_cnt" * 60 / 100));

```

Test Datasets

```

create or replace view retwsp_rf_cust_attr_test_vw as (SELECT v1.* FROM (
-- randomly divide members of the population into subgroups based on target classes
SELECT a.*, row_number() OVER (partition by EDUCATION_BCKGND_CODE ORDER BY
ORA_HASH(CUSTOMER_WID)) "_partition_caseid"
FROM
retwsp_rf_cust_attr_vw a) v1, (
-- get the count of subgroups based on target classes
SELECT
EDUCATION_BCKGND_CODE,
COUNT(*) "_partition_target_cnt" FROM
retwsp_rf_cust_attr_vw GROUP BY EDUCATION_BCKGND_CODE) v2
WHERE v1.EDUCATION_BCKGND_CODE = v2.EDUCATION_BCKGND_CODE
-- random sample subgroups based on target classes in respect to the sample size
AND ORA_HASH(v1."_partition_caseid", v2."_partition_target_cnt"-1, 0) <=
(v2."_partition_target_cnt" * 40 / 100));

```

Explore Data Analysis

Now we have customer reviews we will review, build text to find document frequencies, and display them in charts. We will further extract key text for each review.

Review Customer Behavior Analysis using Text Mining

This script can be found under SQLWorkshop ' SQL Scripts [Text Mining Feature Extraction]

-----Build Text and Token

```

DECLARE
    v_policy_name    VARCHAR2(4000);
    v_lexer_name     VARCHAR2(4000);
BEGIN
    v_policy_name := 'RETWSP_POLICY';
    v_lexer_name  := 'RETWSP_LEXER';
    ctx_ddl.drop_preference(v_lexer_name);
    ctx_ddl.drop_policy(
        policy_name => v_policy_name
    );
    ctx_ddl.create_preference(
        v_lexer_name,
        'BASIC_LEXER'
    );
    ctx_ddl.create_policy(
        policy_name => v_policy_name,
        lexer       => v_lexer_name,
        stoplist    => 'CTXSYS.DEFAULT_STOPLIST'
    );

```

```

END;
/
DROP TABLE retwsp_feature;

CREATE TABLE retwsp_feature
AS
    SELECT
        'REVIEWTEXT' "COLUMN_NAME",
        token,
        value,
        rank,
        count
    FROM
        (
            SELECT
                token,
                value,
                RANK() OVER(
                    ORDER BY value ASC
                ) rank,
                count
            FROM
                (
                    SELECT
                        column_value token,
                        ln(n / COUNT(*) ) value,
                        COUNT(*) count
                    FROM
                        (
                            SELECT
                                t2.column_value,
                                t1.n
                            FROM
                                (
                                    SELECT
                                        COUNT(*) OVER() n,
                                        ROWNUM rn,
                                        odmr_engine_text.dm_policy_tokens(
                                            'RETWSP_POLICY',
                                            reviewtext
                                        ) nt
                                    FROM
                                        retwsp_customer_product_review
                                ) t1,
                                    TABLE ( t1.nt ) t2
                                GROUP BY
                                    t2.column_value,
                                    t1.rn,
                                    t1.n
                            )
                        GROUP BY
                            column_value,
                            n
                    )
                )
        )
    WHERE
        rank <= 3000;

SELECT
    *
FROM
    retwsp_feature;

```

```

create table retwsp_review_df as
SELECT
  *
FROM
  (
    SELECT
      nested_result.attribute_name,
      COUNT(nested_result.attribute_name) count_val
    FROM
      (
        WITH
          /* Start of sql for node: RETWSP_CUSTOMER_PRODUCT_REVIEW */ "N$10001" AS (
            SELECT /*+ inline */
              "RETWSP_CUSTOMER_PRODUCT_REVIEW"."ASIN",
              "RETWSP_CUSTOMER_PRODUCT_REVIEW"."REVIEWERID",
              "RETWSP_CUSTOMER_PRODUCT_REVIEW"."OVERALL",
              "RETWSP_CUSTOMER_PRODUCT_REVIEW"."REVIEWTEXT"
            FROM
              "RETWSP_DEMO_1"."RETWSP_CUSTOMER_PRODUCT_REVIEW"
          )
          /* End of sql for node: RETWSP_CUSTOMER_PRODUCT_REVIEW *//,
          /* Start of sql for node: Build Text */ "N$10002" AS (
            SELECT /*+ inline */
              "REVIEWERID",
              "ASIN",
              "OVERALL",
              odmr_engine_text.dm_text_token_features(
                'RETWSP_POLICY',
                "REVIEWTEXT",
                'RETWSP_FEATURE',
                NULL,
                50,
                'IDF'
              ) "REVIEWTEXT_TOK"
            FROM
              "N$10001"
          )
          /* End of sql for node: Build Text */ SELECT
            *
          FROM
            "N$10002"
          ) output_result,
      TABLE ( output_result.reviewtext_tok ) nested_result
    GROUP BY
      nested_result.attribute_name
  )
ORDER BY count_val DESC;

select * from retwsp_review_df;
-----Feature Extraction and
Feature Comparison -----
-- Create the settings table
DROP TABLE RETWSP_ESA_settings;
CREATE TABLE RETWSP_ESA_settings (
  setting_name VARCHAR2(30),
  setting_value VARCHAR2(30));

DECLARE ----- sub-block begins
  already_exists EXCEPTION;
  PRAGMA EXCEPTION_INIT(already_exists, -00955);

```

```

        v_stmt VARCHAR2(4000);
        v_param_name VARCHAR2(100);
        v_param_value VARCHAR2(100);
BEGIN
    dbms_output.put_line('Start Populate settings table' ||
dbms_data_mining.algo_name);
    dbms_output.put_line('Start Populate settings table' ||
dbms_data_mining.algo_nonnegative_matrix_factor);
    v_param_name := dbms_data_mining.algo_name;
    v_param_value := dbms_data_mining.ALGO_NONNEGATIVE_MATRIX_FACTOR;
    v_stmt := 'INSERT INTO RETWSP_ESA_settings (setting_name, setting_value) VALUES
('' || v_param_name || ''', '' || v_param_value || ''')';
    dbms_output.put_line('Start Populate settings table v_stmt --' || v_stmt);
    EXECUTE IMMEDIATE v_stmt;
    v_param_name := dbms_data_mining.prep_auto;
    v_param_value := dbms_data_mining.prep_auto_on;
    v_stmt := 'INSERT INTO RETWSP_ESA_settings (setting_name, setting_value) VALUES
('' || v_param_name || ''', '' || v_param_value || ''')';
    dbms_output.put_line('Start Populate settings table v_stmt --' || v_stmt);
    EXECUTE IMMEDIATE v_stmt;
    EXCEPTION
        WHEN already_exists THEN
            dbms_output.put_line('Exception not found');
END; ----- sub-block ends
/

create view RETWSP_CUST_PROD_REVIEW_VW as (select reviewid, reviewtext from
RETWSP_CUSTOMER_PRODUCT_REVIEW);

DECLARE
    v_xlst          dbms_data_mining_transform.TRANSFORM_LIST;
    v_policy_name    VARCHAR2(130) := 'RETWSP_POLICY';
    v_model_name     varchar2(50) := 'RETWSP_ESA_MODEL';
BEGIN
    v_xlst := dbms_data_mining_transform.TRANSFORM_LIST();

    DBMS_DATA_MINING_TRANSFORM.SET_TRANSFORM(v_xlst, 'REVIEWTEXT', NULL, 'REVIEWTEXT',
NULL, 'TEXT(POLICY_NAME: ' || v_policy_name || ') (MAX_FEATURES:3000) (MIN_DOCUMENTS:1)
(TOKEN_TYPE:NORMAL)');
    DBMS_DATA_MINING.DROP_MODEL(v_model_name, TRUE);
    DBMS_DATA_MINING.CREATE_MODEL(
        model_name          => v_model_name,
        mining_function      => DBMS_DATA_MINING.FEATURE_EXTRACTION,
        data_table_name      => 'RETWSP_CUST_PROD_REVIEW_VW',
        case_id_column_name  => 'REVIEWID',
        settings_table_name  => 'RETWSP_ESA_SETTINGS',
        xform_list           => v_xlst);
END;
/

-----
-- List top (largest) 3 features that represent are represented in each review.
-- Explain the attributes which most impact those features.
-- This can be used in UI to display all key features in each review.
select REPLACE(xt.attr_name, "REVIEWTEXT.", ''), xt.attr_value, xt.attr_weight,
xt.attr_rank
from (SELECT S.feature_id fid, value val,
        FEATURE_DETAILS(RETWSP_ESA_MODEL, S.feature_id, 5 using T.*) det
FROM
    (SELECT v.*, FEATURE_SET(RETWSP_ESA_MODEL, 3 USING *) fset
    FROM RETWSP_CUSTOMER_PRODUCT_REVIEW v
    WHERE reviewid = 1) T,

```

```

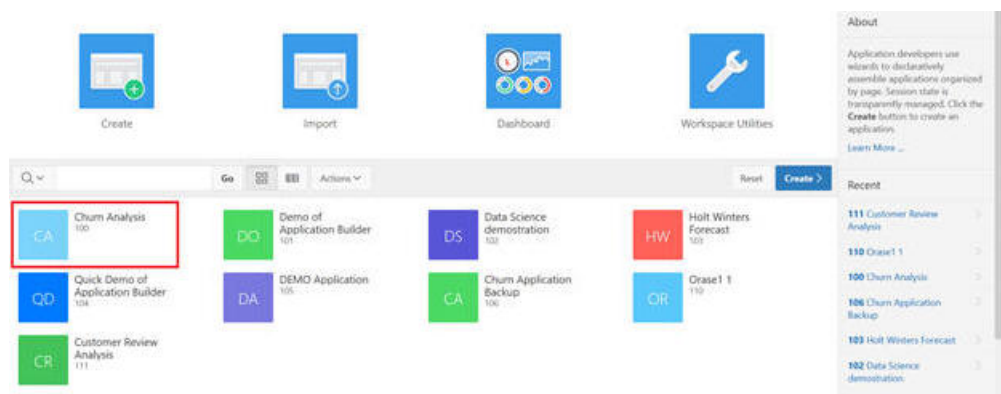
TABLE(T.fset) S
order by val desc) X, XMLTABLE('/Details'
PASSING X.DET
COLUMNS
"ALGORITHM" VARCHAR2(30) PATH '@algorithm',
"FEATURE" VARCHAR2(30) PATH '@feature',
RECORDS XMLTYPE PATH '/Details') R,
XMLTABLE ('/Details/Attribute'
PASSING R.RECORDS
COLUMNS
"ATTR_NAME" VARCHAR2(30) PATH '@name',
"ATTR_VALUE" VARCHAR2(120) PATH '@actualValue',
"ATTR_WEIGHT" VARCHAR2(10) PATH '@weight',
"ATTR_RANK" VARCHAR2(10) PATH '@rank'
) XT
ORDER BY ATTR_RANK;

```

To display the prepared data in UI for end user to review, complete the following:

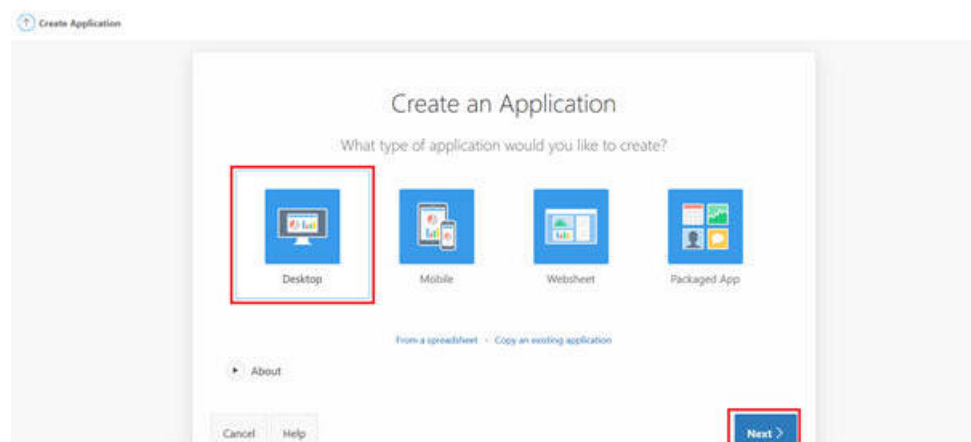
1. Select **Application Builder** to create the UI.

Figure A-33 Churn Analysis



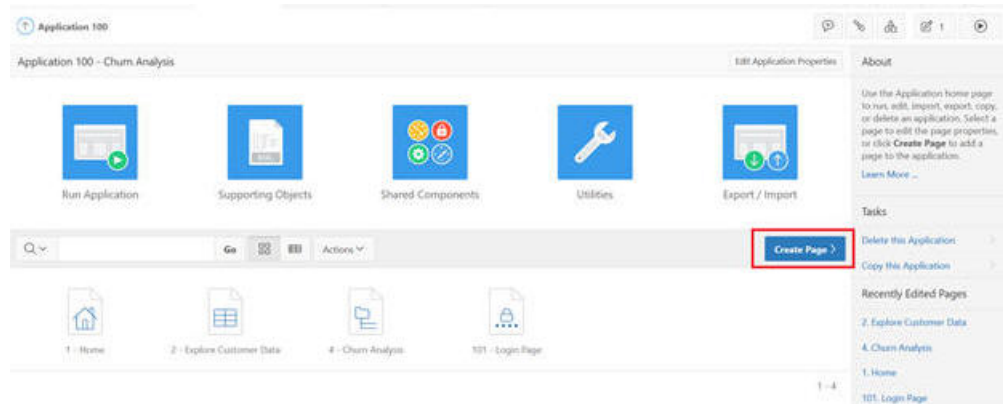
2. Click **Create**.

Figure A-34 Create an Application



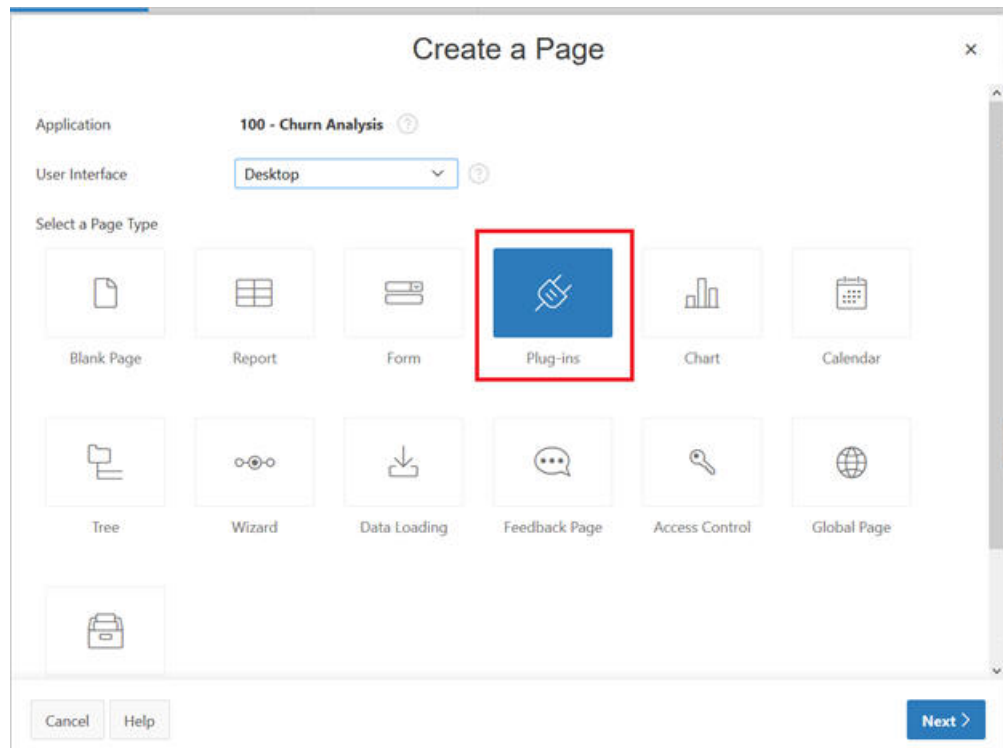
3. Select **Create Page**.

Figure A-35 Create Page



4. Select **Plugin** and click **Next**.

Figure A-36 Plug-In



5. Select **Type**.

Figure A-37 Select Type

Create Page

Type

Plug-in: ☒ Tag Cloud ?

Show Unsupported: Yes No ?

< Cancel Next >

6. Populate Page Name and click **Next**.

Figure A-38 Populate Page Name

Create Page

Page Attributes

Region Type: Tag Cloud [Plug-in]

* Page Number: 5 ?

* Page Name: Document Frequencies ?

* Page Mode: Normal Modal Dialog ?

Breadcrumb: - do not use breadcrumbs on page - ?

< Cancel Next >

7. Select the Navigation Menu entries and click **Next**.

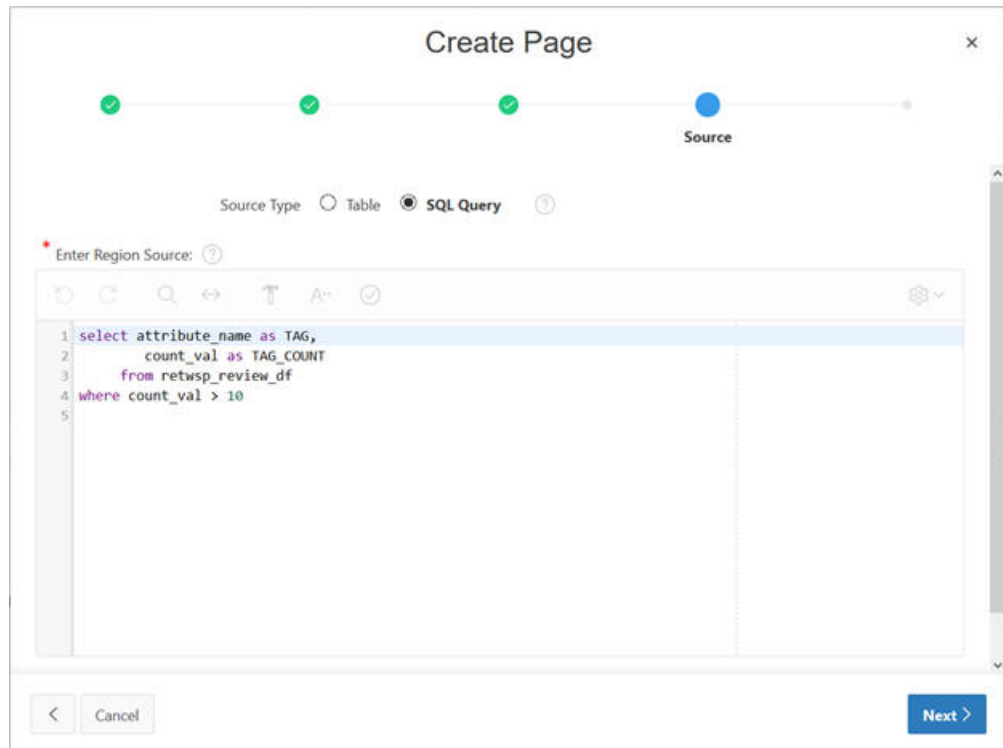
Figure A-39 Navigation Menu

The screenshot shows the 'Create Page' dialog box with the 'Navigation Menu' step selected. The 'Navigation Preference' section has three radio buttons: 'Do not associate this page with a navigation menu entry', 'Create a new navigation menu entry' (which is selected and highlighted with a red box), and 'Identify an existing navigation menu entry for this page'. Below this, the 'New Navigation Menu Entry' field contains the text 'Document Frequencies'. The 'Parent Navigation Menu Entry' dropdown menu is open, showing a list of options: 'No parent selected' (highlighted in blue), 'Home', 'Explore Data', and 'Churn Analysis Results'. At the bottom of the dialog, there are 'Cancel' and 'Next >' buttons.

8. Click **Next** and populate SQL to build the chart.

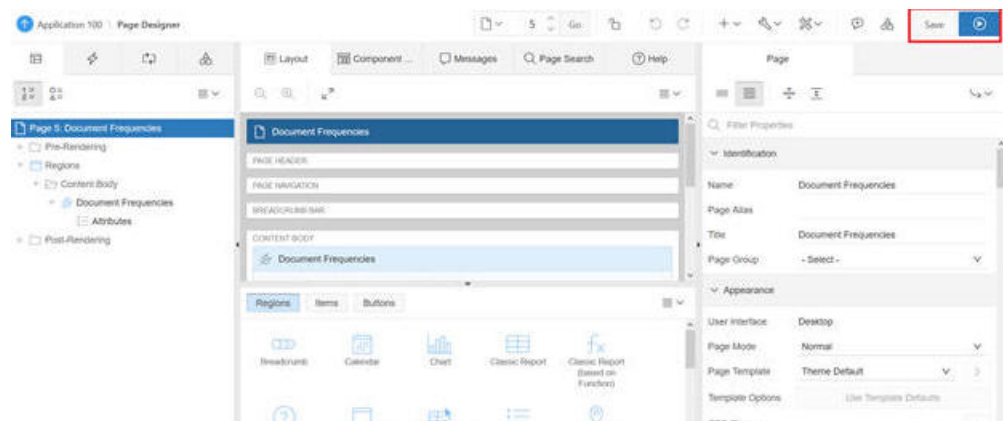
```
select attribute_name as TAG,  
       count_val as TAG_COUNT  
  from retwsp_review_df  
 where count_val > 30  
 order by attribute_name;
```

Figure A-40 SQL Query



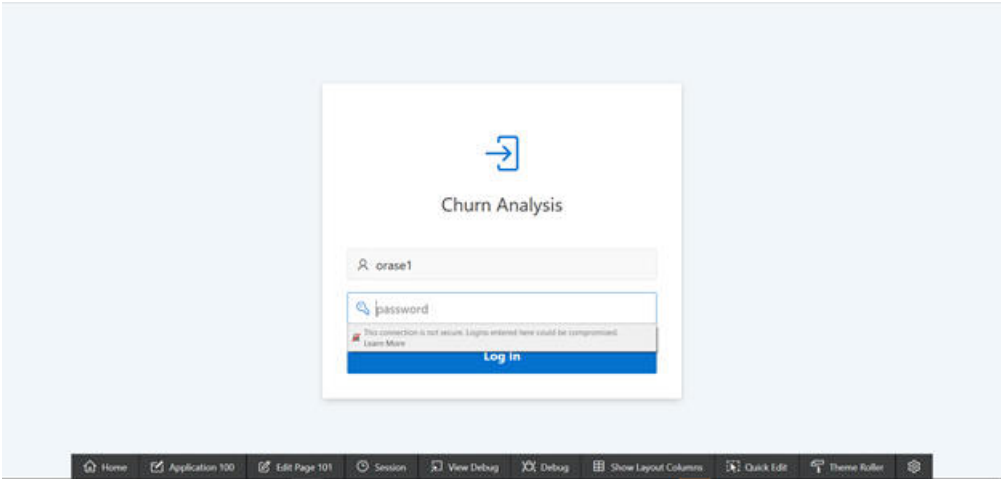
9. You see the Page Designer. Click **Save** and Run icon.

Figure A-41 Page Designer



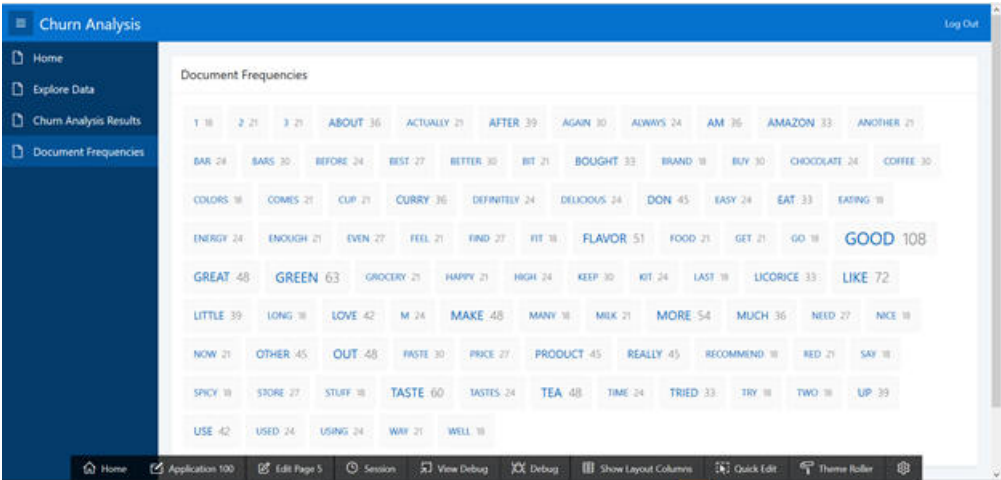
10. Log in with username/password.

Figure A-42 Churn Login



11. The just designed screen is displayed, with the following content.

Figure A-43 Churn Analysis

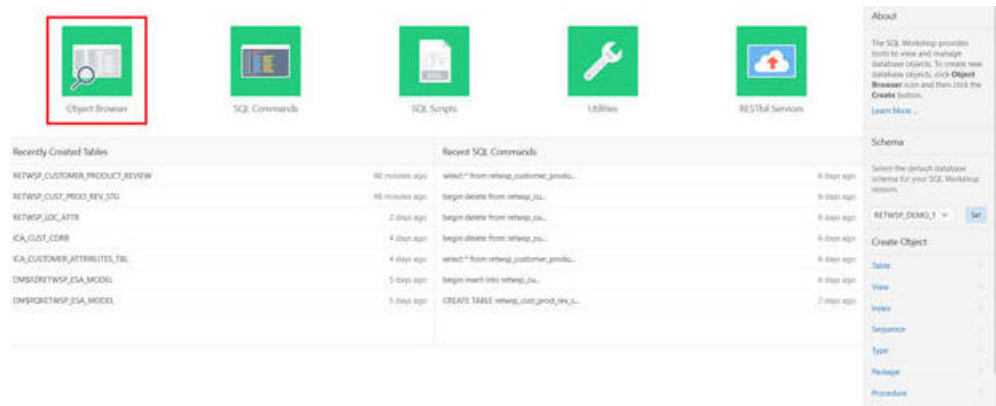


Innovate

Churn Analysis using ODM

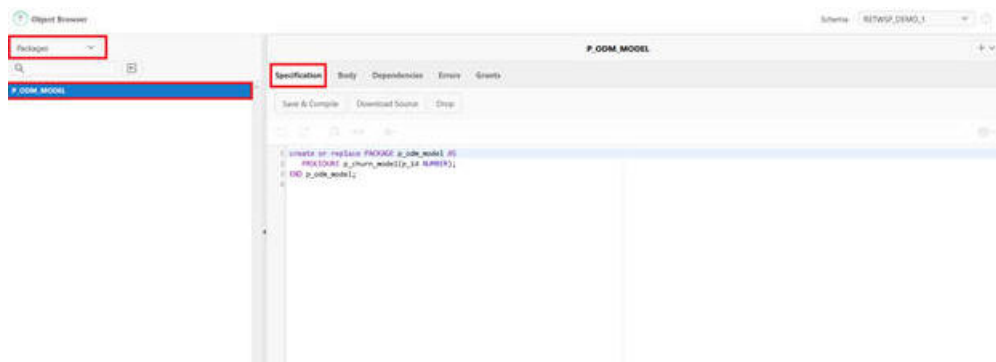
1. View SQL Workshop Object Browser.

Figure A-44 Object Browser



2. Select **Packages** and **Specification**.

Figure A-45 Packages and Specification



3. Select the body that creates a classification model using the Decision Tree algorithm.

Key considerations in the code are

- Note the default classification algorithm in ODM is Naive Bayes. In order to override, create and populate a settings table to be used as input to the model.

Examples of other possible settings are:

(dbms_data_mining.tree_impurity_metric, 'TREE_IMPURITY_ENTROPY')

(dbms_data_mining.tree_term_max_depth, 5)

(dbms_data_mining.tree_term_minrec_split, 5)

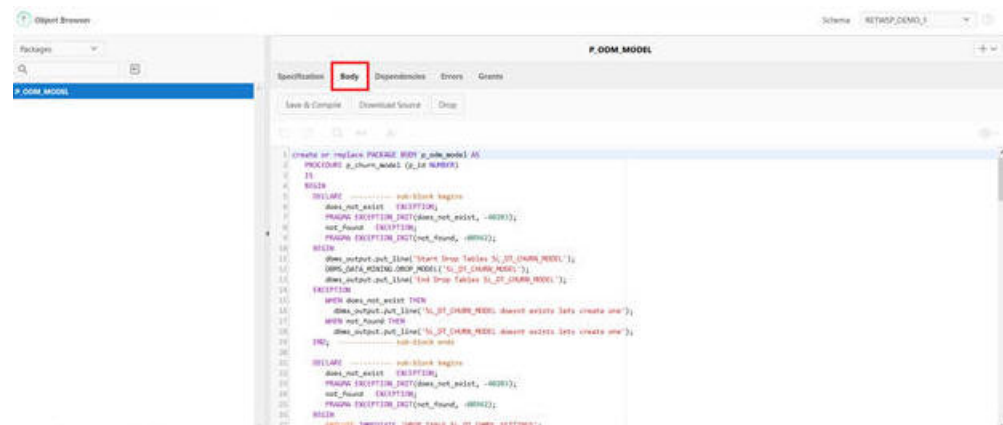
(dbms_data_mining.tree_term_minpct_split, 2)

(dbms_data_mining.tree_term_minrec_node, 5)

(dbms_data_mining.tree_term_minpct_node, 0.05)

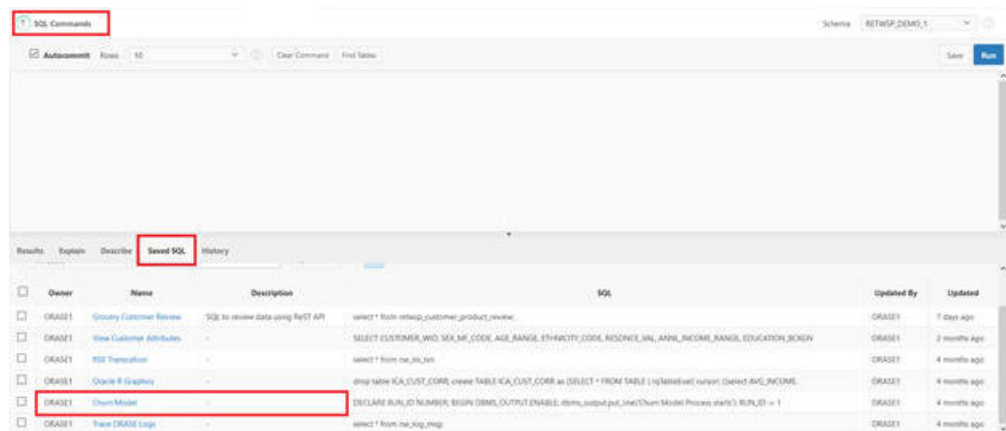
- A cost matrix is used to influence the weighting of mis-classification during model creation and scoring.

Figure A-46 Cost Matrix



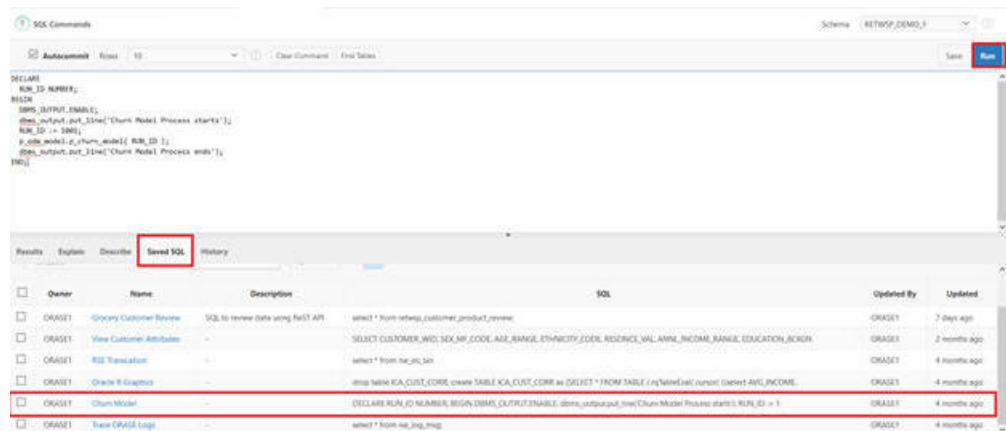
4. Execute the procedure via SQL Command.

Figure A-47 SQL Command



5. Execute Churn Model analysis and view the results.

Figure A-48 Model Analysis Results



Decision Tree Model Details

Here are the results.

```
SELECT
  dbms_data_mining.get_model_details_xml('SL_DT_CHURN_MODEL')
AS DT_DETAILSFROM dual;
```

Results of the XML is as follows, next step will be to parse the XML and

```
<PMML version="2.1">
  <Header copyright="Copyright (c) 2004, Oracle Corporation. All rights reserved."/>
  <DataDictionary numberOfFields="4">
    <DataField name="AGE_RANGE" optype="categorical"/>
    <DataField name="ANNL_INCOME_RANGE" optype="categorical"/>
    <DataField name="CHURN_SCORE" optype="categorical"/>
    <DataField name="PARTY_TYPE_CODE" optype="categorical"/>
  </DataDictionary>
  <TreeModel modelName="SL_DT_CHURN_MODEL" functionName="classification"
splitCharacteristic="binarySplit">
    <Extension name="buildSettings">
      <Setting name="TREE_IMPURITY_METRIC" value="TREE_IMPURITY_GINI"/>
      <Setting name="TREE_TERM_MAX_DEPTH" value="7"/>
      <Setting name="TREE_TERM_MINPCT_NODE" value=".05"/>
      <Setting name="TREE_TERM_MINPCT_SPLIT" value=".1"/>
      <Setting name="TREE_TERM_MINREC_NODE" value="10"/>
      <Setting name="TREE_TERM_MINREC_SPLIT" value="20"/>
    </Extension>
    <costMatrix>
      <costElement>
        <actualValue>0</actualValue>
        <predictedValue>0</predictedValue>
        <cost>0</cost>
      </costElement>
      <costElement>
        <actualValue>0</actualValue>
        <predictedValue>1</predictedValue>
        <cost>1</cost>
      </costElement>
      <costElement>
        <actualValue>0</actualValue>
        <predictedValue>2</predictedValue>
        <cost>2</cost>
      </costElement>
      <costElement>
        <actualValue>0</actualValue>
        <predictedValue>3</predictedValue>
        <cost>3</cost>
      </costElement>
      <costElement>
        <actualValue>1</actualValue>
        <predictedValue>0</predictedValue>
        <cost>1</cost>
      </costElement>
      <costElement>
        <actualValue>1</actualValue>
        <predictedValue>1</predictedValue>
        <cost>0</cost>
      </costElement>
      <costElement>
        <actualValue>1</actualValue>
        <predictedValue>2</predictedValue>
```

```

        <cost>2</cost>
    </costElement>
    <costElement>
        <actualValue>1</actualValue>
        <predictedValue>3</predictedValue>
        <cost>3</cost>
    </costElement>
    <costElement>
        <actualValue>2</actualValue>
        <predictedValue>0</predictedValue>
        <cost>3</cost>
    </costElement>
    <costElement>
        <actualValue>2</actualValue>
        <predictedValue>1</predictedValue>
        <cost>2</cost>
    </costElement>
    <costElement>
        <actualValue>2</actualValue>
        <predictedValue>2</predictedValue>
        <cost>0</cost>
    </costElement>
    <costElement>
        <actualValue>2</actualValue>
        <predictedValue>3</predictedValue>
        <cost>1</cost>
    </costElement>
    <costElement>
        <actualValue>3</actualValue>
        <predictedValue>0</predictedValue>
        <cost>3</cost>
    </costElement>
    <costElement>
        <actualValue>3</actualValue>
        <predictedValue>1</predictedValue>
        <cost>2</cost>
    </costElement>
    <costElement>
        <actualValue>3</actualValue>
        <predictedValue>2</predictedValue>
        <cost>1</cost>
    </costElement>
    <costElement>
        <actualValue>3</actualValue>
        <predictedValue>3</predictedValue>
        <cost>0</cost>
    </costElement>
</costMatrix>
</Extension>
<MiningSchema>
    <MiningField name="AGE_RANGE" usageType="active"/>
    <MiningField name="ANNL_INCOME_RANGE" usageType="active"/>
    <MiningField name="CHURN_SCORE" usageType="predicted"/>
    <MiningField name="PARTY_TYPE_CODE" usageType="active"/>
</MiningSchema>
<Node id="0" score="2" recordCount="100427">
    <True/>
    <ScoreDistribution value="2" recordCount="46479"/>
    <ScoreDistribution value="1" recordCount="32131"/>
    <ScoreDistribution value="3" recordCount="13794"/>
    <ScoreDistribution value="0" recordCount="8023"/>
    <Node id="1" score="2" recordCount="71604">

```

```

    <CompoundPredicate booleanOperator="surrogate">
      <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
        <Array type="string">&quot;20-29&quot; &quot;40-49&quot; &quot;50-59&quot;
&quot;70-79&quot; </Array>
      </SimpleSetPredicate>
      <SimpleSetPredicate field="ANNL_INCOME_RANGE" booleanOperator="isIn">
        <Array type="string">&quot;0k-39k&quot; &quot;40k-59k&quot;
&quot;80k-99k&quot; </Array>
      </SimpleSetPredicate>
    </CompoundPredicate>
    <ScoreDistribution value="2" recordCount="46479"/>
    <ScoreDistribution value="3" recordCount="13794"/>
    <ScoreDistribution value="1" recordCount="11331"/>
    <Node id="2" score="2" recordCount="43586">
      <CompoundPredicate booleanOperator="surrogate">
        <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
          <Array type="string">&quot;40-49&quot; &quot;70-79&quot; </Array>
        </SimpleSetPredicate>
        <SimpleSetPredicate field="PARTY_TYPE_CODE" booleanOperator="isIn">
          <Array type="string">&quot;Cautious Spender&quot; &quot;Mainstream
Shoppers&quot; &quot;Money and Brains&quot; </Array>
        </SimpleSetPredicate>
      </CompoundPredicate>
      <ScoreDistribution value="2" recordCount="29792"/>
      <ScoreDistribution value="3" recordCount="13794"/>
      <Node id="5" score="2" recordCount="41577">
        <CompoundPredicate booleanOperator="surrogate">
          <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
            <Array type="string">&quot;40-49&quot; </Array>
          </SimpleSetPredicate>
          <SimpleSetPredicate field="ANNL_INCOME_RANGE" booleanOperator="isIn">
            <Array type="string">&quot;0k-39k&quot; &quot;40k-59k&quot;
&quot;60k-79k&quot; &quot;80k-99k&quot; </Array>
          </SimpleSetPredicate>
        </CompoundPredicate>
        <ScoreDistribution value="2" recordCount="27783"/>
        <ScoreDistribution value="3" recordCount="13794"/>
      </Node>
      <Node id="6" score="2" recordCount="2009">
        <CompoundPredicate booleanOperator="surrogate">
          <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
            <Array type="string">&quot;70-79&quot; </Array>
          </SimpleSetPredicate>
          <SimpleSetPredicate field="ANNL_INCOME_RANGE" booleanOperator="isIn">
            <Array type="string">&quot;100k+&quot; </Array>
          </SimpleSetPredicate>
        </CompoundPredicate>
        <ScoreDistribution value="2" recordCount="2009"/>
      </Node>
    </Node>
    <Node id="3" score="2" recordCount="28018">
      <CompoundPredicate booleanOperator="surrogate">
        <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
          <Array type="string">&quot;20-29&quot; &quot;50-59&quot; </Array>
        </SimpleSetPredicate>
        <SimpleSetPredicate field="PARTY_TYPE_CODE" booleanOperator="isIn">
          <Array type="string">&quot;Livin Large&quot; &quot;Value Seeker&quot;
&quot;Young Professional&quot; </Array>
        </SimpleSetPredicate>
      </CompoundPredicate>
      <ScoreDistribution value="2" recordCount="16687"/>
      <ScoreDistribution value="1" recordCount="11331"/>
    </Node>
  </Node>

```



```

<Node id="7" score="2" recordCount="15567">
  <CompoundPredicate booleanOperator="surrogate">
    <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
      <Array type="string">&quot;20-29&quot; </Array>
    </SimpleSetPredicate>
    <SimpleSetPredicate field="ANNL_INCOME_RANGE" booleanOperator="isIn">
      <Array type="string">&quot;0k-39k&quot; &quot;80k-99k&quot; </Array>
    </SimpleSetPredicate>
  </CompoundPredicate>
  <ScoreDistribution value="2" recordCount="10398"/>
  <ScoreDistribution value="1" recordCount="5169"/>
</Node>
<Node id="8" score="2" recordCount="12451">
  <CompoundPredicate booleanOperator="surrogate">
    <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
      <Array type="string">&quot;50-59&quot; </Array>
    </SimpleSetPredicate>
    <SimpleSetPredicate field="ANNL_INCOME_RANGE" booleanOperator="isIn">
      <Array type="string">&quot;100k+&quot; &quot;40k-59k&quot; </Array>
    </SimpleSetPredicate>
  </CompoundPredicate>
  <ScoreDistribution value="2" recordCount="6289"/>
  <ScoreDistribution value="1" recordCount="6162"/>
</Node>
</Node>
<Node id="4" score="1" recordCount="28823">
  <CompoundPredicate booleanOperator="surrogate">
    <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
      <Array type="string">&quot;30-39&quot; &quot;60-69&quot; </Array>
    </SimpleSetPredicate>
    <SimpleSetPredicate field="ANNL_INCOME_RANGE" booleanOperator="isIn">
      <Array type="string">&quot;100k+&quot; &quot;60k-79k&quot; </Array>
    </SimpleSetPredicate>
  </CompoundPredicate>
  <ScoreDistribution value="1" recordCount="20800"/>
  <ScoreDistribution value="0" recordCount="8023"/>
  <Node id="9" score="1" recordCount="16772">
    <CompoundPredicate booleanOperator="surrogate">
      <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
        <Array type="string">&quot;60-69&quot; </Array>
      </SimpleSetPredicate>
      <SimpleSetPredicate field="PARTY_TYPE_CODE" booleanOperator="isIn">
        <Array type="string">&quot;Livin Large&quot; &quot;Value Seeker&quot; </
Array>
      </SimpleSetPredicate>
    </CompoundPredicate>
    <ScoreDistribution value="1" recordCount="16772"/>
  </Node>
  <Node id="10" score="0" recordCount="12051">
    <CompoundPredicate booleanOperator="surrogate">
      <SimpleSetPredicate field="AGE_RANGE" booleanOperator="isIn">
        <Array type="string">&quot;30-39&quot; </Array>
      </SimpleSetPredicate>
      <SimpleSetPredicate field="PARTY_TYPE_CODE" booleanOperator="isIn">
        <Array type="string">&quot;Mainstream Shoppers&quot; &quot;Young
Professional&quot; </Array>
      </SimpleSetPredicate>
    </CompoundPredicate>
    <ScoreDistribution value="0" recordCount="8023"/>
    <ScoreDistribution value="1" recordCount="4028"/>
  </Node>

```

```

        </Node>
    </Node>
</TreeModel>
</PMML>

```

Parse XML and insert data into a table <RETWSP_TREE_CHURN_RULES> using SQL.

```

Create table RETWSP_TREE_CHURN_RULES as
WITH X as
(SELECT * FROM
XMLTable('for $n in /PMML/TreeModel//Node
    let $rf :=
        if (count($n/CompoundPredicate) > 0) then
            $n/CompoundPredicate/*[1]/@field
        else
            if (count($n/SimplePredicate) > 0) then
                $n/SimplePredicate/@field
            else
                $n/SimpleSetPredicate/@field
    let $ro :=
        if (count($n/CompoundPredicate) > 0) then
            if ($n/CompoundPredicate/*[1] instance of
                element(SimplePredicate)) then
                $n/CompoundPredicate/*[1]/@operator
            else if ($n/CompoundPredicate/*[1] instance of
                element(SimpleSetPredicate)) then
                ("in")
            else ()
        else
            if (count($n/SimplePredicate) > 0) then
                $n/SimplePredicate/@operator
            else if (count($n/SimpleSetPredicate) > 0) then
                ("in")
            else ()
    let $rv :=
        if (count($n/CompoundPredicate) > 0) then
            if ($n/CompoundPredicate/*[1] instance of
                element(SimplePredicate)) then
                $n/CompoundPredicate/*[1]/@value
            else
                $n/CompoundPredicate/*[1]/Array/text()
        else
            if (count($n/SimplePredicate) > 0) then
                $n/SimplePredicate/@value
            else
                $n/SimpleSetPredicate/Array/text()
    let $sf :=
        if (count($n/CompoundPredicate) > 0) then
            $n/CompoundPredicate/*[2]/@field
        else ()
    let $so :=
        if (count($n/CompoundPredicate) > 0) then
            if ($n/CompoundPredicate/*[2] instance of
                element(SimplePredicate)) then
                $n/CompoundPredicate/*[2]/@operator
            else if ($n/CompoundPredicate/*[2] instance of
                element(SimpleSetPredicate)) then
                ("in")
            else ()
        else ()
    let $sv :=
        if (count($n/CompoundPredicate) > 0) then

```

```

        if ($n/CompoundPredicate/*[2] instance of
            element(SimplePredicate)) then
            $n/CompoundPredicate/*[2]/@value
        else
            $n/CompoundPredicate/*[2]/Array/text()
        else ()
    return
    <pred id="{ $n/./@id}"
        score="{ $n/@score}"
        rec="{ $n/@recordCount}"
        cid="{ $n/@id}"
        rf="{ $rf}"
        ro="{ $ro}"
        rv="{ $rv}"
        sf="{ $sf}"
        so="{ $so}"
        sv="{ $sv}"
    />'
    passing dbms_data_mining.get_model_details_xml('SL_DT_CHURN_MODEL')
    COLUMNS
        parent_node_id    NUMBER PATH '/pred/@id',
        child_node_id     NUMBER PATH '/pred/@cid',
        rec                NUMBER PATH '/pred/@rec',
        score              VARCHAR2(4000) PATH '/pred/@score',
        rule_field         VARCHAR2(4000) PATH '/pred/@rf',
        rule_op            VARCHAR2(20) PATH '/pred/@ro',
        rule_value         VARCHAR2(4000) PATH '/pred/@rv',
        surr_field         VARCHAR2(4000) PATH '/pred/@sf',
        surr_op            VARCHAR2(20) PATH '/pred/@so',
        surr_value         VARCHAR2(4000) PATH '/pred/@sv'))
    select pid parent_node, nid node, rec record_count,
        score prediction, rule_pred local_rule, surr_pred local_surrogate,
        rtrim(replace(full_rule,'$O$D$M$'),' AND') full_simple_rule from (
    select row_number() over (partition by nid order by rn desc) rn,
    pid, nid, rec, score, rule_pred, surr_pred, full_rule from (
    select rn, pid, nid, rec, score, rule_pred, surr_pred,
        sys_connect_by_path(pred, '$O$D$M$') full_rule from (
    select row_number() over (partition by nid order by rid) rn,
    pid, nid, rec, score, rule_pred, surr_pred,
        nvl2(pred,pred || ' AND ',null) pred from(
    select rid, pid, nid, rec, score, rule_pred, surr_pred,
        decode(rn, 1, pred, null) pred from (
    select rid, nid, rec, score, pid, rule_pred, surr_pred,
        nvl2(root_op, '(' || root_field || ' ' || root_op || ' ' || root_value || ')',
    null) pred,
        row_number() over (partition by nid, root_field, root_op order by rid desc) rn from
    (
    SELECT
        connect_by_root(parent_node_id) rid,
        child_node_id nid,
        rec, score,
        connect_by_root(rule_field) root_field,
        connect_by_root(rule_op) root_op,
        connect_by_root(rule_value) root_value,
        nvl2(rule_op, '(' || rule_field || ' ' || rule_op || ' ' || rule_value || ')',
    null) rule_pred,
        nvl2(surr_op, '(' || surr_field || ' ' || surr_op || ' ' || surr_value || ')',
    null) surr_pred,
        parent_node_id pid
    FROM (
        SELECT parent_node_id, child_node_id, rec, score, rule_field, surr_field,
        rule_op, surr_op,

```

```

        replace(replace(rule_value,'" "', '""', '""'),''','', '','') rule_value,
        replace(replace(surr_value,'" "', '""', '""'),''','', '','') surr_value
FROM (
    SELECT parent_node_id, child_node_id, rec, score, rule_field, surr_field,
        decode(rule_op, 'lessOrEqual', '<=', 'greaterThan', '>', rule_op) rule_op,
        decode(rule_op, 'in', '('||rule_value||')', rule_value) rule_value,
        decode(surr_op, 'lessOrEqual', '<=', 'greaterThan', '>', surr_op) surr_op,
        decode(surr_op, 'in', '('||surr_value||')', surr_value) surr_value
    FROM X)
)
CONNECT BY PRIOR child_node_id = parent_node_id
)
)
)
CONNECT BY PRIOR rn = rn - 1
        AND PRIOR nid = nid
START WITH rn = 1
)
)
where rn = 1;

```

This sql is also listed in SQL Scripts.

Figure A-49 SQL Listed SQL Script



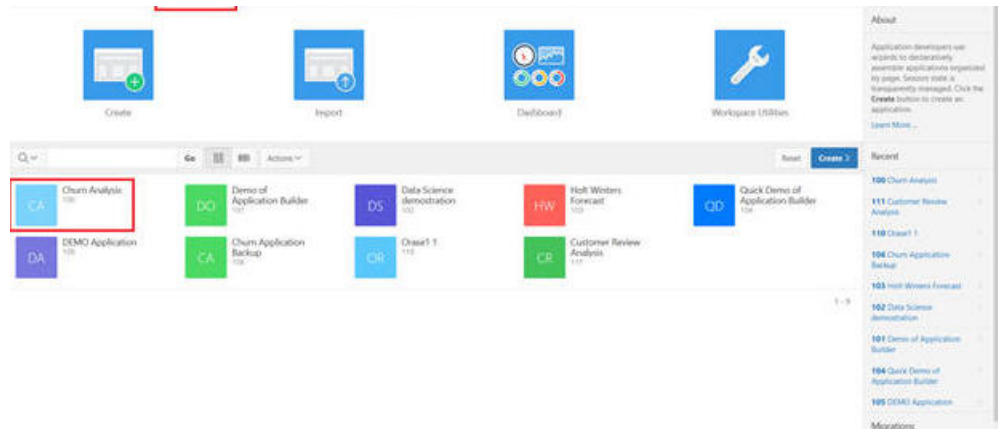
	Edit	Owner	Name	Created	Updated By	Updated	Bytes	Results	Run
☐		ORASE1	Pearl Decision Tree Model ML	1 second ago	ORASE1	1 second ago	6,206	0	
☐		ORASE1	Explore Data Correlation Graphs R-Graphviz	2 months ago	ORASE1	4 days ago	380	6	
☐		ORASE1	Text Mining Feature Extraction	5 days ago	ORASE1	5 days ago	6,253	0	

Save the churn rules by creating table RETWSP_TREE_CHURN_RULES as (<sql above>).

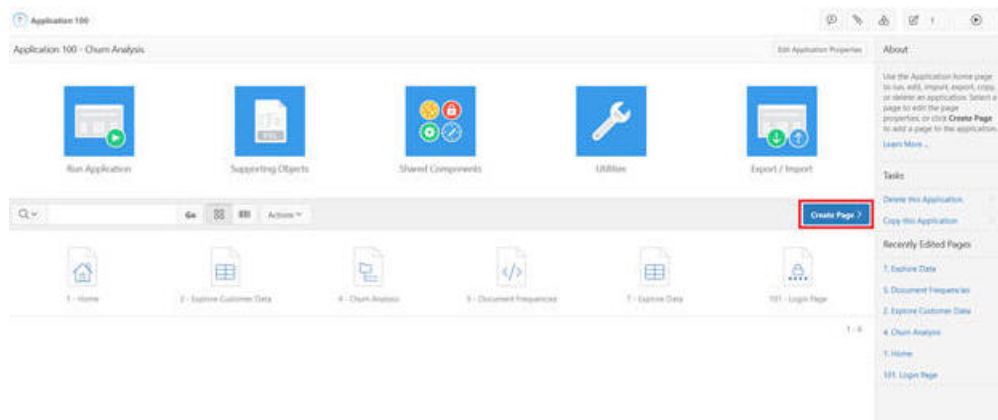
Display Decision Tree Rules

Create an application to display these rules.

1. Select **Churn Analysis** from App Builder.

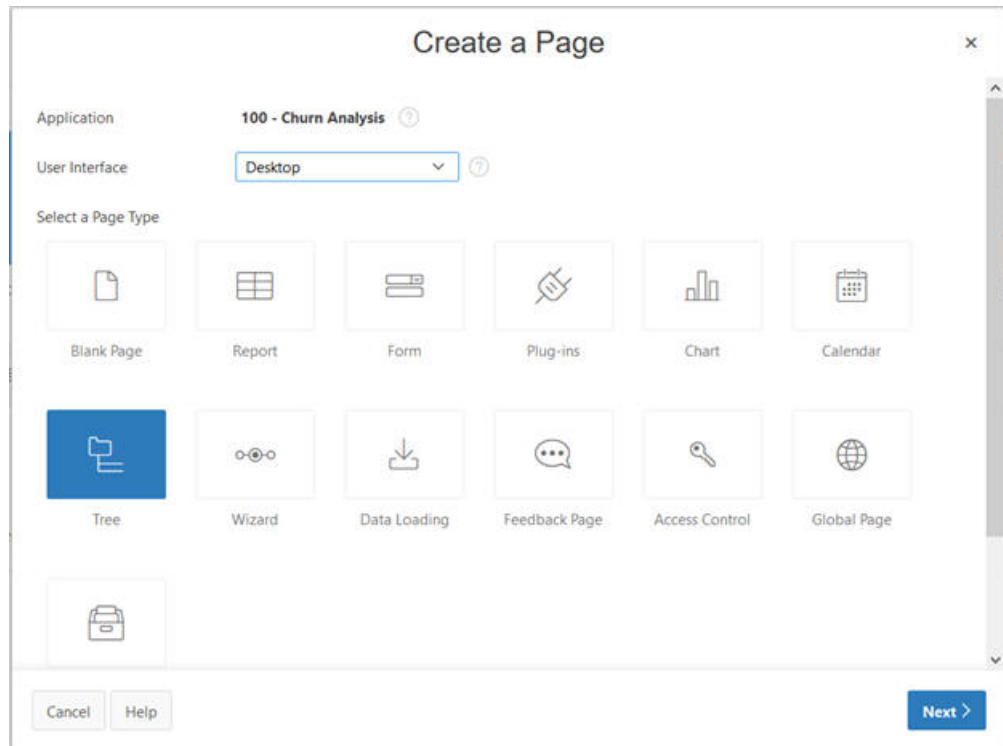
Figure A-50 Select Churn Analysis

2. Create a page.

Figure A-51 Create Page

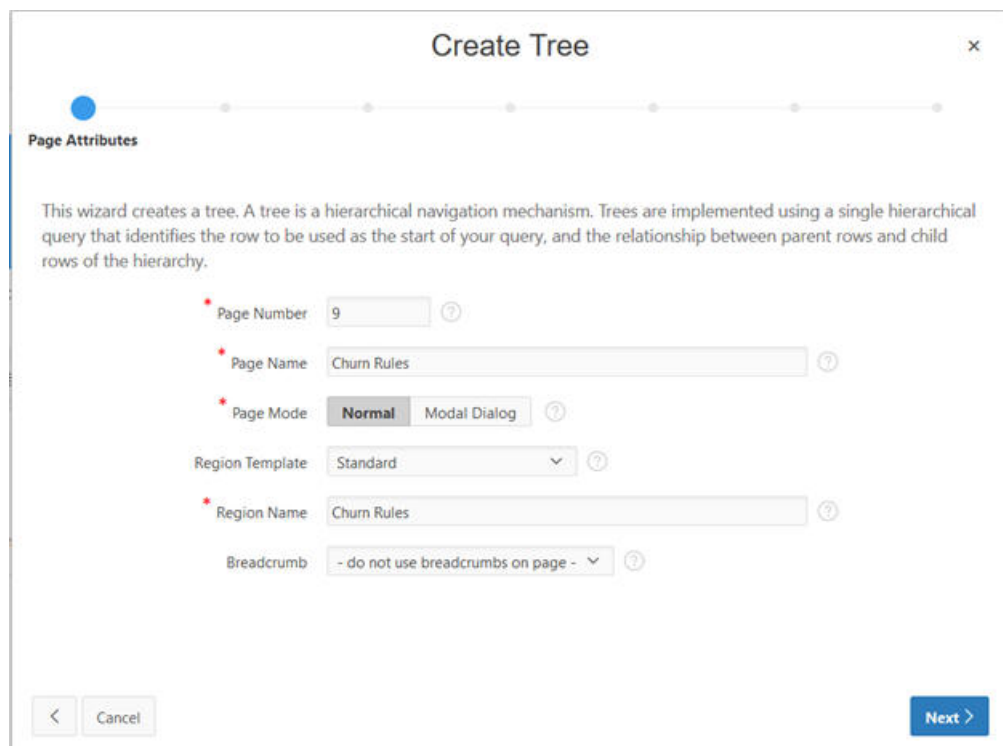
3. Select the Tree to display.

Figure A-52 Select Tree



4. Populate Page Name as Churn Rules and click **Next**.

Figure A-53 Page Attributes



5. Select **Create a new navigation menu entry**.

Figure A-54 New Navigation Menu Entry

The screenshot shows the 'Create Tree' dialog box with a progress bar at the top. The first step, 'Navigation Menu', is highlighted with a blue dot. Below the progress bar, the 'Navigation Preference' section has two radio buttons: 'Do not associate this page with a navigation menu entry' and 'Create a new navigation menu entry', which is selected and highlighted with a red rectangle. The 'New Navigation Menu Entry' field contains 'Churn Rules'. The 'Parent Navigation Menu Entry' dropdown menu is open, showing options: '- No parent selected -', 'Home', 'Explore Data', 'Churn Analysis Results', 'Document Frequencies', and 'Explore Data'. At the bottom, there are 'Cancel' and 'Next >' buttons.

6. Select **RETWSP_TREE_CHURN_RULES** from Table/View name list.

Figure A-55 Table/View Name

The screenshot shows the 'Create Tree' dialog box with a progress bar at the top. The second step, 'Table / View Owner and Name', is highlighted with a blue dot. Below the progress bar, the text 'Select the owner of the table or view from which you want to draw the tree query.' is displayed. The 'Table / View Owner' dropdown menu is set to 'RETWSP_DEMO_1'. The 'Table / View Name' dropdown menu is set to 'RETWSP_TREE_CHURN_RULES (table)' and is highlighted with a red rectangle. At the bottom, there are 'Cancel' and 'Next >' buttons.

Figure A-56 Tree Query

Create Tree [X]

Progress: 1. Select Table (✓) 2. Select Columns (✓) 3. Select Parent Column (✓) 4. **Query** (●)

A tree is based on a query and returns data that can be represented in a hierarchy. A **start with .. connect by** clause will be used to generate the hierarchical query for your tree. Use this page to identify the column you want to use as the ID, the Parent ID, and text that should appear on the nodes. The Start With column will be used to specify the root of the hierarchical query, and its value can be based on an existing item, static value or SQL query returning a single value.

ID: NODE (Number) [?] **Parent ID**: PARENT_NODE (Number) [?] **Node Text**: LOCAL_RULE (Varchar2) [?] **Start With**: PARENT_NODE (Number) [?] **Start Tree**: Value is NULL [?]

> **Example Start With Query**

[<] [Cancel] [Next >]

Figure A-57 Where Clause

Create Tree [X]

Progress: 1. Select Table (✓) 2. Select Columns (✓) 3. Select Parent Column (✓) 4. Query (✓) 5. **Where and Order by** (●)

Identify an optional where clause and order siblings by column for your query.

Where Clause (for example ename = 'JONES') [?] **Order Siblings By** (for example ENAME): PREDICTION (Varchar2) [?]

> **Current Query**

[<] [Cancel] [Next >]

Figure A-58 Tree Attributes

Create Tree

Identify the button, tooltip and link attributes you want to define on your tree. To make leaf node text a link, select **Existing Application Item**.

Include Buttons: ☒ Collapse All ☒ Expand All

Selected Node Page Item

Tooltip: None

Link Option: ☒ Nothing ☐ Existing Application Item

< Cancel Next >

Figure A-59 Tree Confirmation

Create Tree

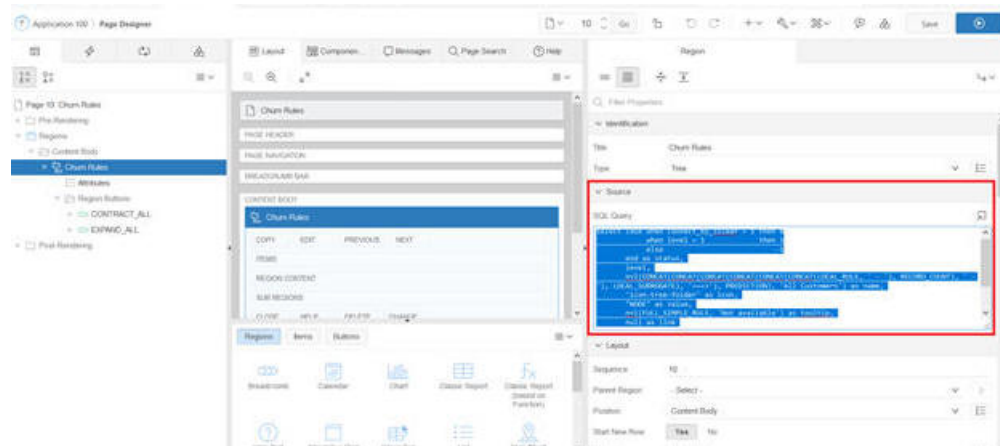
You have requested to create a tree page with the following attributes. Please confirm your selections.

Application	100
Page	9
Page Name	Churn Rules
Region Title	Churn Rules
Region Template	Standard

< Cancel Create

7. You see Page Designer.

Figure A-60 Page Designer

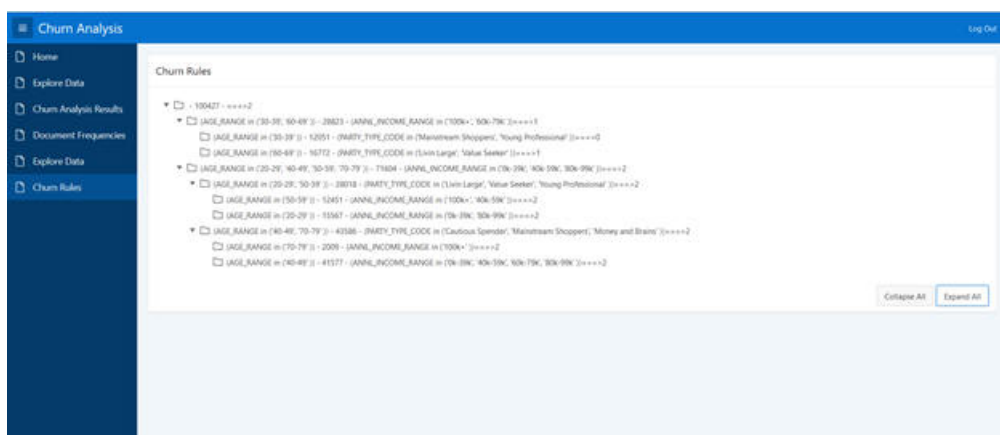


8. Replace the following query to better format and execute the run.

```
select case when connect_by_isleaf = 1 then 0
           when level = 1 then 1
           else -1
        end as status,
        level,
        nvl(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(LOCAL_RULE, ' - '),
        RECORD_COUNT), ' - '), LOCAL_SURROGATE), '==>'), PREDICTION), 'All Customers') as
        name,
        'icon-tree-folder' as icon,
        "NODE" as value,
        nvl(FULL_SIMPLE_RULE, 'Not available') as tooltip,
        null as link
from "#OWNER#". "RETWSP_TREE_CHURN_RULES"
start with "PARENT_NODE" is null
connect by prior "NODE" = "PARENT_NODE"
order siblings by "RECORD_COUNT"
```

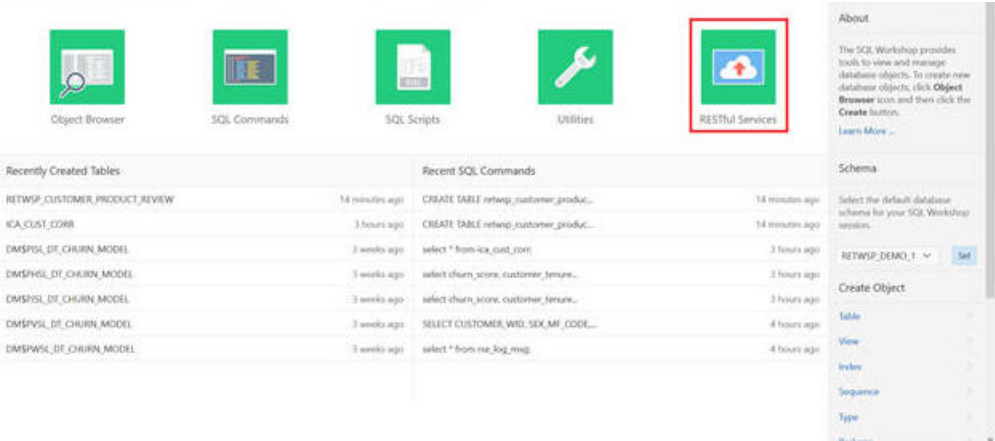
9. Run a new window

Figure A-61 Run New Window



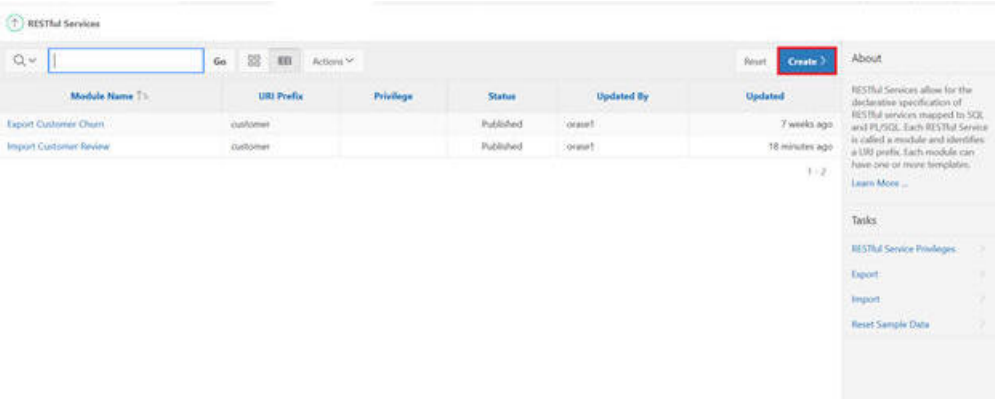
1. View RESTful Services
2. Create ReST API, which you can use to load customer reviews in the above table.

Figure A-62 ReST API



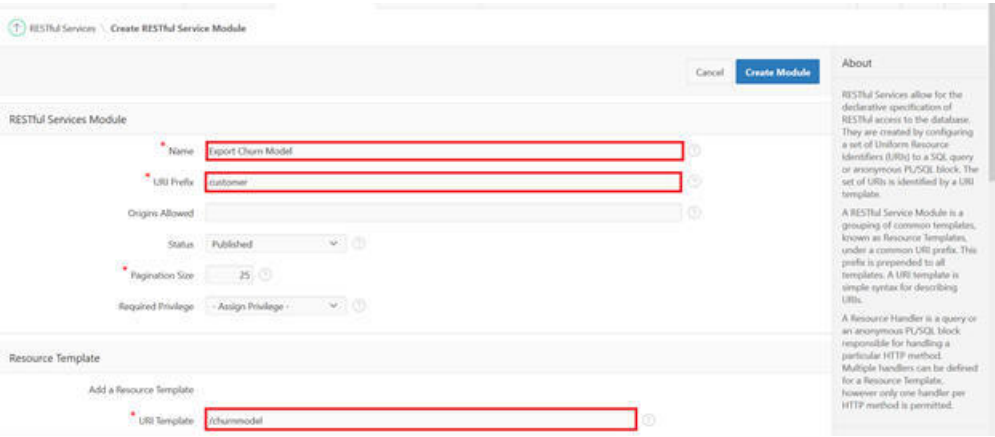
3. Click **Create**.

Figure A-63 Create



4. Populate with Name, URI Prefix, and URI Template.

Figure A-64 Populate ReST Fields



5. Select **GET** and populate the SQL as:

```
SELECT PARENT_NODE,
       NODE,
       RECORD_COUNT,
       PREDICTION,
       LOCAL_RULE,
       LOCAL_SURROGATE,
       FULL_SIMPLE_RULE
FROM RETWSP_TREE_CHURN_MODEL
```

Figure A-65 Get

6. Select **GET** and then click **Test**.

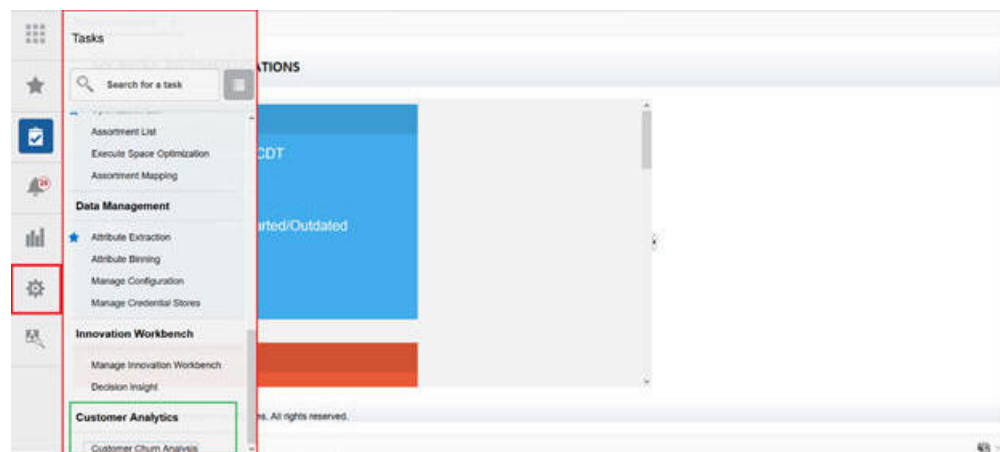
Figure A-66 Click Test

7. Data is displayed in the window and any request to this end point `http://<url>` will return the data shown in [Figure A-67](#) in JSON format.

[illegible]

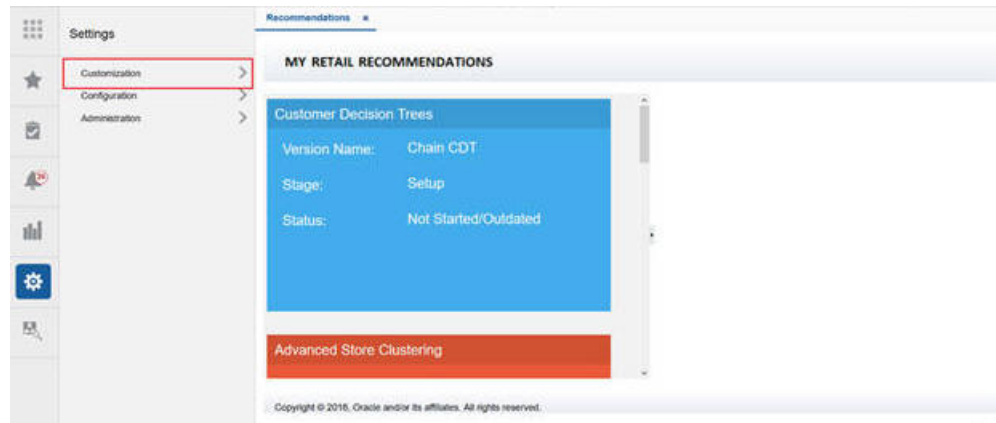
This exercise illustrates how to add a link to the application Task Navigation.

- ### Figure A-68 Tasks



- Implementation Guide
G5396-02
Copyright © 2026. Oracle and/or its affiliates.

Figure A-69 Customization

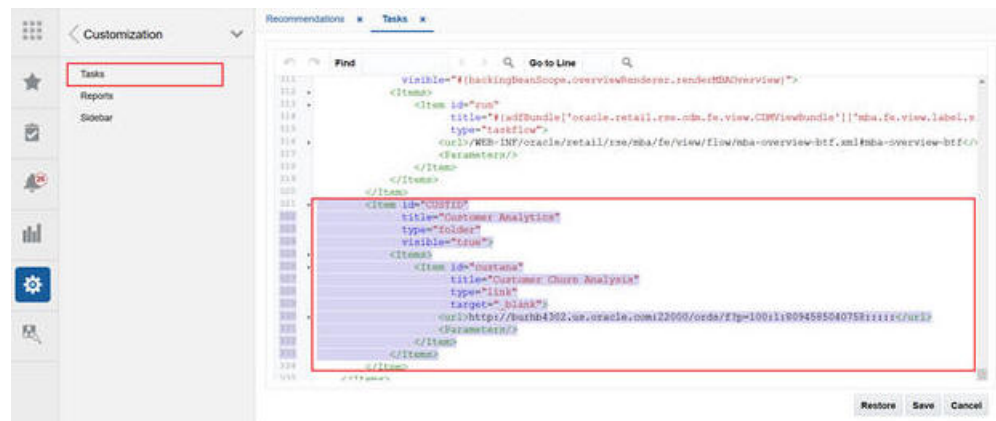


- The Content Area displays Navigation XML. At the end of the XML, add the following content (see Customer Analytics). Update the highlighted contents.

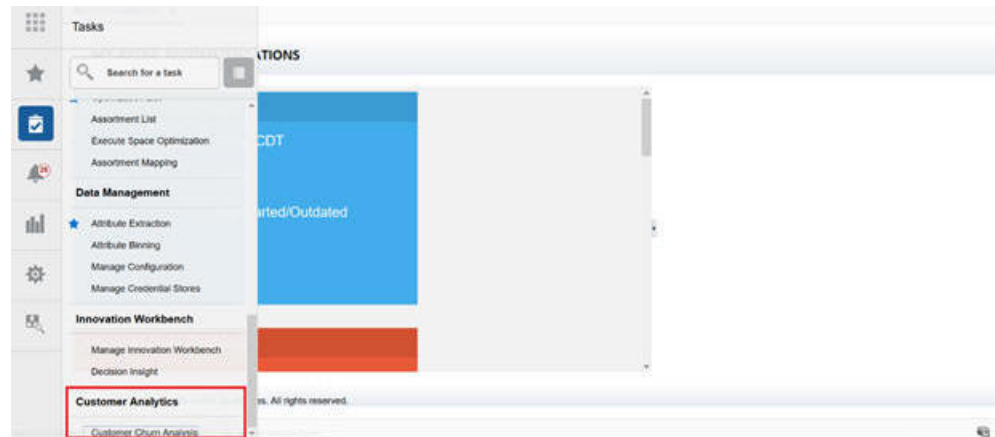
```

    <Item id="CUSTID"                title="Customer Analytics"
    type="folder"                    visible="true">                                <Items>                                <Item
    id="custana"                      title="Customer Churn
    Analysis"                          type="link"
    target="_blank">                                <url>http://<url>/ords/f?
    p=100:1:8094585040758::::</url>                                <Parameters/>                                </
    Item>                                </Items>                                </Item>
  
```

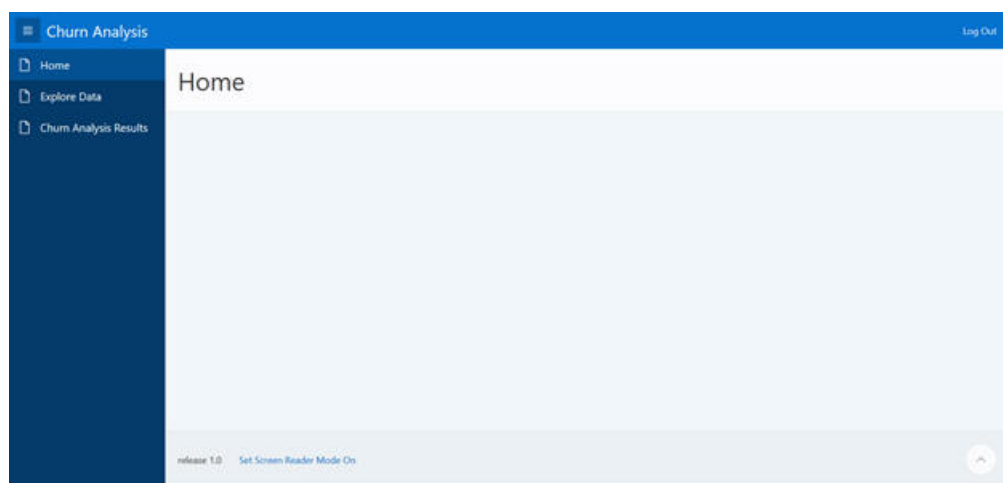
Figure A-70 Highlighted SQL



- Click **Save**. Log out and then log back in. You should see:

Figure A-71 After Logout

5. Click the Customer Churn Analysis link system. A new tab is launched in the browser for the end user.

Figure A-72 New Tab

Database Tools

How to Allocate Privileges

None of the tables in Retailer Workspace Schema are accessible in the Application Schema.

Executing queries in the Application Schema will result in an error. Until and unless access is provided, the user will not be able to access object/data in another schema.

```
describe retwsp_demo_1.p_churn_model;
ERROR:
ORA-04043: object retwsp_demo_1.p_churn_model does not exist
```

```
select * from RETWSP_TREE_CHURN_RULES;
ORA-00942: table or view does not exist
00942. 00000 - "table or view does not exist"
*Cause:
```

*Action:
Error at Line: 1 Column: 15

How to Execute a Job using DBMS Scheduler

Long running jobs must be executed under RETAILER_WORKSPACE_JOBS. These are executed in resource groups dedicated to the retailer schema.

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
  job_name => 'retwsp_churn_model',
  job_type => 'STORED_PROCEDURE',
  job_action => 'retwsp.pkg_customer_analytics.proc_churn_model',
  start_date => '01-JAN-17 07.00.00 PM US/Pacific',
  repeat_interval => 'FREQ=YEARLY; BYDATE=0331,0630,0930,1231; ',
  end_date => '31-DEC-17 07.00.00 PM US/Pacific',
  job_class => 'RETAILER_WORKSPACE_JOBS',
  comments => 'Retailer workspace churn model job');
END;
/
```


B

Appendix: Option Forecast Details

Options Forecast is a module within Oracle Retail Artificial Intelligence Foundation (AIF) services, developed to provide advanced analytical support for retail assortment planning. This module leverages Oracle Retail's robust AI-driven analytics capabilities to transform vast amounts of historical sales data, item lifecycle information, and product attributes into actionable insights for retailers. By examining which attributes and characteristics are associated with high-performing products, Options Forecast helps identify patterns and preferences that drive successful sales outcomes.

The module delivers a comprehensive set of outputs, including Attribute Weights, Optimized History, Total Option Count, Option Count by Attribute Value, and Sales Potential. These outputs help retailers understand which product features most influence customer demand and how historical trends can inform future assortment selections. In addition, Options Forecast performs granular analysis at the store level, evaluating the number of units sold, pinpointing lost opportunities and sales, and highlighting instances of missed or delayed deliveries.

Through these advanced analytics, Options Forecast equips retailers with precise, data-driven recommendations for assortment planning. By identifying growth opportunities and reducing the risk of missed sales, the module enables organizations to strategically optimize their product offerings for upcoming seasons, ultimately supporting better business outcomes and competitive advantage in the retail marketplace.

The Options Forecast methodology within AIF automates key aspects of the Assortment Planning solution by providing strategic recommendations for assortment strategy. This methodology is structured around a systematic run setup, comprising the following steps:

- Life Cycle: Analyzing the complete lifecycle of products to understand performance trends.
- Optimize History: Leveraging historical sales data to identify patterns and inform future decisions.
- Attribute Weights: Determining the relative importance of product attributes that drive sales.
- Incrementality: Assessing the incremental value each product brings to the overall assortment.
- Sales Potential: Estimating the future sales potential of various assortment options.

Through this structured approach, Options Forecast enhances the effectiveness and precision of assortment planning decisions. This appendix provides details for the parameters involved in the calculations of each of these steps.

Life Cycle

The life cycle of an item is calculated by identifying when the item first started selling (the start date) and when it stopped selling (the end date) using historical sales data for each store or location. By calculating the number of weeks the item was available for sale, this process helps determine how long each item was actively selling, which is useful for planning future assortments.

Table B-1 Life Cycle Parameters

Parameter Name	Parameter Description	Input Specifications (Default, Range, and so on)	Parameter Explanation (AIF)
OPT_LC_EOL_CHECK_WEEKS	Number of Weeks to Check for End of Life Promos/ Stockouts	Default: 4 (weeks)	Specifies the number of weeks to consider when evaluating end of life promotions or end of life stock outs. For example, if it is set to 8, the system will look at the sales data for the last 8 weeks of an item's life cycle to check if the item had significantly more promotions or stock outs than previous weeks.
OPT_LC_EOL_MIN_SO_PERCENT	Percentage Threshold to Consider End of Life Stock Outs	Default: 0.3 Range: 0.0 - 1.0	Specifies the minimum percentage to consider that week in the end of life window as a stocked-out. An item is considered heavily stocked out when the average percentage of store count with inventory (percentage calculated wrt max store count with inventory over entire life) is higher than this threshold (for example, for default, 30%).
OPT_LC_ITEMS_STILL_SELLING_FLG	Still Selling Filter	Y/N Default: Y	Drop items that are still selling. An item is considered still selling if it has at least one positive sale in the last OPT_LC_EOL_PAST_N_WEEKS weeks from the end of the training period. Y means consider this filter in rule extraction and N means do not apply this filter.
OPT_LC_ITEMS_WITH_FEW_SLS_FLG	Few Sales Filter	Y/N Default: Y	Y: Drop items with less than OPT_LC_MIN_NUM_POS_SLS_WKS sales data points (weeks). N: Do Not drop it. Y means consider this filter in rule extraction and N means do not apply this filter.
OPT_LC_LOW_SELLERS_FLG	Low Sellers Filter	Y/N Default: N Y: Drop items that are low sellers. N: Do not apply filter.	This flag controls whether to exclude items that are low sellers. Specifically, these are items that sell less than one unit per week (fixed value) on average over all weeks in their time series. If Y, items that meet this condition are excluded from the exit date estimation process.

Table B-1 (Cont.) Life Cycle Parameters

Parameter Name	Parameter Description	Input Specifications (Default, Range, and so on)	Parameter Explanation (AIF)
OPT_LC_MIN_NUM_POS_S LS_WKS	Minimum Number of Weeks with Non-zero Sales	Default: 5 (weeks)	Minimum number of non-zero sales weeks an item must have to calculate the lifespan. Items with less than this value are not considered for exit dates rules generation.
OPT_LC_RECENCY_IN_WK S	Item Recency Minimum Weeks	Applied with OPT_LC_RECENT_ITEMS_ FLG. Default: 8 (weeks) For example, if an item has had its first sale in the past eight weeks, exclude from exit date estimation.	Specifies the number of weeks to consider when determining whether an item is recent. Items that have their first sale within this number of weeks from the end of the training period are considered recent and may be excluded from the analysis if OPT_LC_RECENT_ITEMS_ FLG is Y.
OPT_LC_RECENT_ITEMS_ FLG	Recent Items Filter	Y/N Default: Y Y: Drop items that started selling recently, that is, that started selling less than OPT_LC_RECENCY_IN_WK S weeks from the end of the training period (for example, 7 weeks ago). N: Do not apply filter.	Controls whether to exclude items that have recently been introduced. If Y, items that have their first sale within OPT_LC_RECENCY_IN_WK S weeks from the end of the training period are excluded from the exit date estimation process.
OPT_LC_RETURN_SLS_FL G	Only Returns Filter	Y/N Default: N Y: Drop sales entries of items if they are exclusively returns. N: Do not apply filter.	Controls whether to exclude sales entries that are exclusively returns, as returns can artificially extend the lifespan of an item.
OPT_LC_SLS_AFT_98_PRC _FLG	98% Cumulative Percentage Filter	Y/N Default: Y Y: Exit date is estimated as the week when the cumulative sales reach 98% of the total sales. N: Exit date is estimated as the last week with sales.	Determines the method used to estimate the exit date for an item. If Y, the exit date is estimated as the week when the cumulative sales reach 98% of the total sales. If N, the exit date is estimated as the last week with sales.
OPT_LC_SLS_BFR_FST_R CPT_FLG	Sales Before First Receipt Filter	Y/N Default: Y Y: Sales that occur before the first receipt week ID are excluded from exit date estimation. N: Do not apply filter.	Controls whether to exclude sales that occurred before the first receipt week ID. For example, if an item was first received in week 10, and OPT_LC_SLS_BFR_FST_R CPT_FLG is Y, any sales data for that item prior to week 10 will be excluded from the analysis.

Optimize History

Optimized History is a careful process that helps retailers get a true picture of product demand by correcting sales data for issues related to inventory, such as stockouts (times when products were out of stock) and limited variety on the shelves. This allows retailers to answer important questions such as, “What would our sales have looked like if we always had enough of each product available?”

Optimized History is computed both at the item (stylecolor) and attribute level in two separate algorithms. This allows AIF to produce predictions for both how a specific stylecolor would optimally sell, and how any item possessing a certain attribute (that is, being of color grey, or brand Nike) would sell.

Using this data, a statistical model is trained to understand overall customer demand and how much sales are affected when inventory is limited, for each attribute or item. This model helps adjust both the reported sales and the number of product options (option count), reflecting what would have happened if inventory was never an issue. The result is a much clearer and more accurate foundation for planning future assortments, allowing retailers to better decide how much variety in color, style, or material they should carry, even when planning for products that have not been sold before. This ensures future predictions and assortment choices are based on real customer preferences, not just what happened to be in stock in the past.

Table B-2 Optimize History Parameters

Parameter Name	Parameter Description	Input Specifications (Default, Range, and so on)	Parameter Explanation (AIF)
OPT_OH_ITEMS_MISSING_SCWI_FLG	Missing Store Count with Inventory Filter	Y/N filter for data cleaning. Default = Y Y: Drop items with more than OPT_OH_ITEMS_MISSING_SCWI_PCT% having zero or null store_count_with_inv values. N: Do not apply filter.	This flag controls whether to exclude items with a high percentage of missing or zero Store Count With Inventory (SCWI) values. If Y, items with more than OPT_OH_ITEMS_MISSING_SCWI_PCT percentage of weeks having zero or null SCWI values are excluded, indicating that they may have unreliable inventory data and may not be suitable for optimize history corrections.
OPT_OH_ITEMS_MISSING_SCWI_PCT	Missing Store Count with Inventory Percentage	Continuous filter for data cleaning, paired with OPT_OH_ITEMS_MISSING_SCWI_FLG. Range: 0.0-1.0, Default: 0.2 A higher threshold value (such as 0.5 or 50%) will result in more lenient filtering, including items with a larger percentage of missing or zero store_count_with_inv values.	Maximum percentage of sales weeks where store count with inventory (store_count_with_inv column in pmo_options_activities) can be either zero or null. Default is 20%, which means that items with more than 20% of sales weeks having zero or null store_count_with_inv values will be excluded.

Table B-2 (Cont.) Optimize History Parameters

Parameter Name	Parameter Description	Input Specifications (Default, Range, and so on)	Parameter Explanation (AIF)
OPT_OH_ITEMS_STILL_SELLING_FLG	Still Selling Filter	Y/N filter for data cleaning. Y: Drop items that are still selling (drop items with sales in the last OPT_LC_EOL_PAST_N_WK weeks). N: Do not drop items that are still selling. Default = N	This flag controls whether to exclude items that are still actively selling. If Y, items with sales in the last OPT_LC_EOL_PAST_N_WK weeks are excluded, indicating that they may not have reached the end of their life cycle and may not be suitable for optimize history corrections.
OPT_OH_ITEMS_WITH_FEW_SLS_FLG	Few Sales Filter	Y/N filter for data cleaning. Y: Drop items with less than OPT_LC_MIN_NUM_POS_SLS_WKS weeks where sales are positive, in which the default is OPT_LC_MIN_NUM_POS_SLS_WKS=5 weeks. N: Do not drop items meeting this condition. Default = N	This flag controls whether to exclude items with limited sales data. If Y, items with fewer than OPT_LC_MIN_NUM_POS_SLS_WKS weeks of positive sales are excluded, indicating that they may not have sufficient data for reliable optimize history calculations.
OPT_OH_MAX_WKS_APPLY_ED_RULE	Maximum Weeks to Apply Exit Date Rule	Default = 4 For example, with OPT_OH_MAX_WKS_APPLY_ED_RULE=4, if the estimated exit week is 9 weeks beyond the expected life cycle (defined here as start week + weeks to exit), the rule will not be applied because 9 > 4.	Maximum allowed difference in weeks between the calculated exit date and actual life of an item. If the difference is higher than OPT_OH_MAX_WKS_APPLY_ED_RULE, exit date rules are not used to define this item's end of life in the optimize history estimation phase. This parameter controls the tolerance for the difference between the estimated exit week ID and the actual life of the item. If the difference is too large, the extracted exit date rule is not applied, indicating that the rule may not be reliable for optimize history corrections.

Table B-2 (Cont.) Optimize History Parameters

Parameter Name	Parameter Description	Input Specifications (Default, Range, and so on)	Parameter Explanation (AIF)
OPT_OH_RECENT_ITEMS_FLG	Recent Items Filter	Y/N filter for data cleaning. Y: Drop items that started selling recently, that is, that started selling less than OPT_LC_RECENCY_IN_WKS ago (for example, 7 weeks ago). N: Do not drop it. Default = N	This flag controls whether to exclude items that have recently been introduced. If Y, items that have their first sale within OPT_LC_RECENCY_IN_WKS weeks from the end of the training period are excluded, indicating that they may not have sufficient historical data for reliable optimize history calculations.
OPT_OH_ATTR_CORR_THR_ESHOLD	Optimize History Model Type	Range: 0.0 - 1.0 Default: 0.9	Threshold of sales percentage to compute Optimize History by Attribute corrections. 90% means that for a given Attribute, top Attribute Values that cumulatively contribute 90% of the total sales will be corrected.

Attribute Weights

Attribute weight refers to a numerical value assigned to each product attribute, representing its relative importance in determining product similarity within a retail assortment. Products are typically described by several attributes, such as brand, color, collar type, neckline, and sleeve length. However, these attributes do not all contribute equally when assessing how similar two products are. For example, brand and color may have a greater influence on perceived similarity compared to collar type or sleeve length. Attribute weights serve to quantify this difference in importance by assigning higher weights to more influential attributes.

Incrementality

An incrementality curve is a curve that models projected sales across a range of option counts within an assortment. This curve is generated by simulating sales outcomes using estimated arrival rates (which reflect seasonality), product attractiveness scores, and store-specific assortment data. The incrementality curve illustrates the incremental sales lift associated with each additional option introduced to the assortment. By depicting the relationship between the number of available options and the corresponding incremental sales, this curve enables retailers to evaluate the marginal benefit of assortment expansion and supports data-driven assortment planning decisions.

Table B-3 Incrementality Parameters

Parameter Name	Parameter Description	Input Specifications (Default, Range, and so on)	Parameter Explanation (AIF)
OPT_IC_MARKET_SHARE	Market Share	Default = 0.5 Range: 0.0-1.0	Specify the retailer market share, which is the fraction of all potential customers expected to make a purchase when all products are available. Lower market share means more customers choose not to purchase; higher market share means more customers shop with the retailer.

Sales Potential

Sales potential is an estimated measure of the expected sales that a product or assortment can achieve within a specific period. This metric leverages factors such as historical sales data, product attributes, and product category to forecast likely sales outcomes. Primarily, it models information on product sales and attributes, as in color, brand, and so on, to understand how well certain attributes, as well as combinations of attributes, sell. Sales potential helps retailers identify high-opportunity products or attribute combinations and support more accurate, data-driven planning.

Table B-4 Sales Potential Parameters

Parameter Name	Parameter Description	Input Specifications (Default, Range, and so on)	Parameter Explanation (AIF)
SP_MAX_ATTR_VAL_DOMINATION	Maximum Attribute Value Domination	Range: 0.0 - 1.0 Default: 0.5	Specifies the maximum percentage of items an attribute value can dominate for a given attribute to be included in training. For example, if all items have a single attribute value, that attribute is not useful for model training. A value of 0.5 means that if an attribute value is present in more than 50% of the items, the attribute will be dropped.
SP_MIN_ATTR_FREQUENCY	Minimum Attribute Frequency	Range: Integer value ≥ 1 Default: 1	Specifies the minimum number of items that must have a non-null value for an attribute to be included in training. For instance, if only three items have sleeve-length populated, then sleeve-length will be dropped from model training if SP_MIN_ATTR_FREQUENCY is greater than 3.

Table B-4 (Cont.) Sales Potential Parameters

Parameter Name	Parameter Description	Input Specifications (Default, Range, and so on)	Parameter Explanation (AIF)
SP_MIN_ATTR_VAL_DIVERSITY	Minimum Attribute Value Diversity	Range: Integer value ≥ 2 (recommended) Default: 2	Specifies the minimum number of unique attribute values required for an attribute to be included in training. Attributes with low cardinality (that is, fewer unique values) may not be useful for model training.
SP_MIN_LIFECYCLE_WINDOW	Minimum Lifecycle Window	Range: Integer value ≥ 1 Default: 1	Specifies the minimum historical lifecycle length in weeks for an item to be included in training. Items with a short lifecycle may not provide enough data for effective model training.
SP_MIN_MAX_STORE_CNT_WITH_INV	Minimum Value for Max Store Count with Inventory	Range: Integer value ≥ 1 Default: 1	Specifies the minimum value for the maximum store count with inventory for an item to be included in training. Items with low inventory levels across stores may not be representative of the overall sales pattern.
SP_MIN_SALES_UNITS	Minimum Sales Units	Range: Integer value ≥ 1 Default: 1	Specifies the minimum amount of sales units required for an item to be included in training. Items with very low sales may not provide enough data for effective model training.

Glossary

AA

Affinity Analysis.

AC

Advanced Clustering, also known as CIS.

AIF

AI Foundation, which generally refers to the AIF applications and user interfaces.

AIF APPS

The AIF applications job schedule in Process Orchestration & Monitoring (POM).

AIF DATA

The AIF data warehouse job schedule in Process Orchestration & Monitoring (POM).

BI

Business Intelligence.

CDT

Customer Decision Tree.

DB

Database.

DT

Demand Transference.

MDS

Oracle MetaData Services.

POG Hierarchy

Defined by three levels: POG department, POG category, and POG subcategory. The POG hierarchy is used to organize POGs within POG sets. For example, a leaf to root path in the POG hierarchy: Grocery -> Snacks -> Crackers.

POG Node

A leaf node (POG subcategory) within a POG set.

RI

Retail Insights, formerly known as Retail Analytics. Shares the same AIF data warehouse tables as the rest of the platform.

RADM

Retail Analytics Data Model, also known as RI Schema or AIF data warehouse.

RCM

Oracle Retail Category Management.

IPO-DF

Inventory Planning Optimization Demand Forecasting, formerly known as RDF.

Seasonal Attribute

Refers to a specific year independent time period for an APO assortment and a POG set. Examples include Spring, holiday, back to school, year-round. (Also, Season Attribute.)