# Oracle® Retail Advanced Science Engine

Implementation Guide

Release 14.1

**E59126-02**

December 2014

Oracle Retail Advanced Science Engine Implementation Guide, Release 14.1

E59126-02

Primary Author: Judith Meskill

Primary Author: Jay Cummings

**Value-Added Reseller (VAR) Language**

**Oracle Retail VAR Applications**

Applications. For purposes of this section, “alteration” refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

# Contents

## 4 ORASE Common Functionality

## 5 Configuration

## 6 Data Integration and Interfaces

## 7 Market Basket Analysis Overview

## 8 Server Configuration

## 9 Attribute Processing

## A Sample Taskflow_MultiSolution.xml File for ASO/Fusion Client Integration

## B Database Detail Definitions

## C Log File Locations

## D Sample Category Management to ASO Import Files

## E Retail Analytics Interfaces Files

## F Sample Application Context File (DT)

## Glossary of Acronyms

## List of Figures

## List of Tables

# Send Us Your Comments

Oracle® Retail Advanced Science Engine Implementation Guide, Release 14.1

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 Implementation Guides provide detailed information useful for implementing and configuring the application. It helps you to understand the behind-the-scenes processing of the application.

## Audience

This guide is intended for users who configure and use ORASE. It is essential that someone implementing ORASE be experienced with the following:

- Installing, configuring, and managing the Oracle WebLogic application server software and security.
- Installing, configuring, and managing the Oracle relational database management system. The implementer must be familiar with the Database Administrator (DBA) level commands and tasks.
- Installing, configuring, and managing the distributed client/server applications on UNIX-based systems and networks.

## 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.

## Related Documents

For more information, see the following documents in the Oracle Retail Advanced Science Engine documentation set:

- *Oracle Retail Advanced Science Engine Installation Guide*
- *Oracle Retail Advanced Science Engine Release Notes*
- *Oracle Retail Advanced Science Engine Security Guide*
- *Oracle Retail Advanced Science Engine User Guide*

- *Oracle Retail Assortment and Space Optimization User Guide*

## 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

## Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 14.1) or a later patch release (for example, 14.1.1). If you are installing the base release or additional patches, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

## Improved Process for Oracle Retail Documentation Corrections

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

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail 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 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 Documentation on the Oracle Technology Network

Documentation is packaged with each Oracle Retail product release. Oracle Retail product documentation is also 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. These documents are packaged with released code, or you can obtain them through My Oracle Support.)

Documentation should be available on this Web site within a month after a product release.

## Conventions

The following text conventions are used in this document:

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

# 1 Introduction

Oracle Retail Advanced Science Engine (ORASE) is the centralized science engine that supports retail business processes by driving the analytics for both the Oracle Retail Advanced Science Engine (ORASE) and for the Oracle Retail Assortment and Space Optimization (ASO).

ORASE performs data mining and develops analytical parameters to support business processes in Oracle Retail Category Management (CM), Oracle Retail Demand Forecasting (RDF), and Oracle Retail Analytics (RA). It is comprised of the following modules:

- Customer Decision Tree (CDT)
- Demand Transference (DT)
- Advanced Clustering (AC)
- Market Basket Analysis (MBA)

ASO provides a way for planners to inform their decisions about assortment rationalization and to perform Micro Space Optimization.

This implementation guide addresses the implementation of ORASE and ASO.

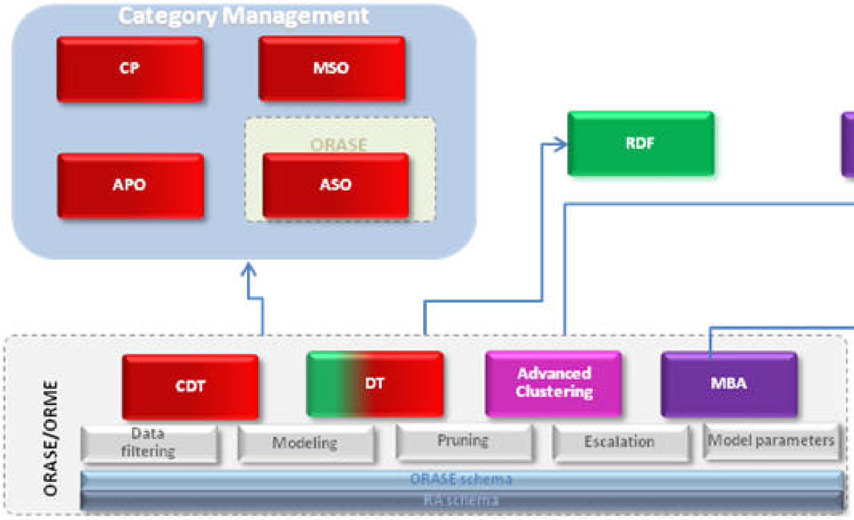
Implementers should be familiar with application servers, the installation process, Oracle databases, system and IT administration, RPAS applications, UNIX commands, including shell configurations and scripts, UNIX directory operations, and symlinks.

## ORASE Overview

ORASE is architected in a modular fashion to serve as a centralized science engine supporting multiple solutions. Figure 1–1 and the discussion that follows illustrate the interaction among the various components that comprise ORASE. The four applications shown in the figure within CM are:

- Category Planning (CP)
- Assortment Planning and Optimization (APO)
- ASO
- Market Basket Analysis

*Figure 1–1 Oracle Retail Advanced Science Engine (ORASE) Application Integration*



ORASE provides science-based functional extensions to CM, RDF, and RA. These three applications provide data inputs to ORASE and ASO and receive data outputs (Clusters, CDTs, and DT models) from ORASE. ASO also generates optimized assortments and planograms (POGs) and replenishment updates. ASO results are not passed as output, but may be extracted from the ORASE database, if desired.

CM is a category planning, assortment planning, and assortment rationalization tool. CM uses Clusters, CDTs, and DT models from ORASE as well as space-optimized assortments from ASO to determine which categories should be carried in a store, how much space should be allocated to a category, which items should be in a category assortment, and how much space should be allocated to an item.

RDF is a statistical and promotional forecasting solution. With the introduction of DT models from ORASE, RDF is able to forecast demand change when an item is either removed or added to an assortment.

RA is a suite of retail-enterprise-level fully integrated Oracle BI applications. ORASE provides RA clusters to analyze sales, inventory levels, promotions and customer data, and market basket analysis to calculate product affinity relationships.

## Integration with Oracle Retail Analytics

ORASE has its own intra-ETL (extraction, transform, and load) ability that reads from RA or the RA Data Model (RADM). ORASE leverages data from RADM via direct schema reads and must be co-deployed on the same database. Note that ORASE does not require the RA BI product, only RADM. In addition, ORASE has its own schema that has been optimized for the analytical processing required for its science modules.

The Oracle Retail Merchandising System (RMS)/RA ETL is available to ORASE retailers, so that they can load ORASE data from RMS into RADM and the ORASE schema. See the RA and RMS documentation sets for more information on the ETL associated with RMS and RA.

## Common Workflow

The ORASE solutions have a similar workflow and user interface (UI). The workflow lets users implement new science modules using similar techniques. For example, a retailer who uses Demand Transference and the Customer Decision Tree 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 modules.

The *Oracle Retail Advanced Science Engine User Guide* and the *Oracle Retail Assortment and Space Optimization User Guide* provide details about using each of these applications.

# ORASE Overview

This section provides an overview of each of the ORASE modules.

## Customer Decision Tree

Customer Decision Trees (CDT), with their dynamic hierarchical structure, help retailers gain insights into the customer decision process. CDT results illustrate a prioritization or importance of specific product attributes that determined the customer’s purchase.

The product includes new science that mines retailer data to understand customer behavior and preferences across multiple channels to develop CDTs. The solution provides insights into what attributes are driving customer purchases. This CDT generation process can be further informed by retailer business insights around attribute prioritization or supplier CDTs.

## Demand Transference

Demand Transference (DT) refers to the shifting or transfer of demand among items within an assortment, as items are added to or deleted from the assortment.

Demand Transference science mines retailer data to identify demand transference effects, which are then used within CM and RDF to drive plans and forecasts informed by planned assortment changes. ASO uses the results from DT to predict the effects on demand of similar SKUs as SKUs are dropped or added to an assortment.

## Advanced Clustering

Advanced Clustering (AC) builds store clusters with similar consumer demand patterns and integrates those clusters into solutions such as assortment planning, category management, pricing, promotion, allocation, and the supply chain.

AC also groups like stores, items, and entities, based on sales volume, profit margin, store format, customer type, demand profile, and promotional effectiveness.

## Market Basket Analysis

Market Basket Analysis (MBA) employs data mining to provide insight into the correlation among products in a customer's basket. Prepackaged integration sends Market Basket Analysis outputs to RA.

# ASO Overview

ASO provides a way for planners to make decisions about optimized assortments. It takes as its input the collection of planograms and the assortments that are mapped to the planograms across a set of stores. A planogram is a collection of fixtures (shelves, pegboards, freezer cabinets) of various lengths. Stores may be grouped together into clusters that share some user-defined characteristics. ASO provides the user with the means to optimize the assortment and the space allocated to it to meet a variety of business goals.

# 2 Architecture

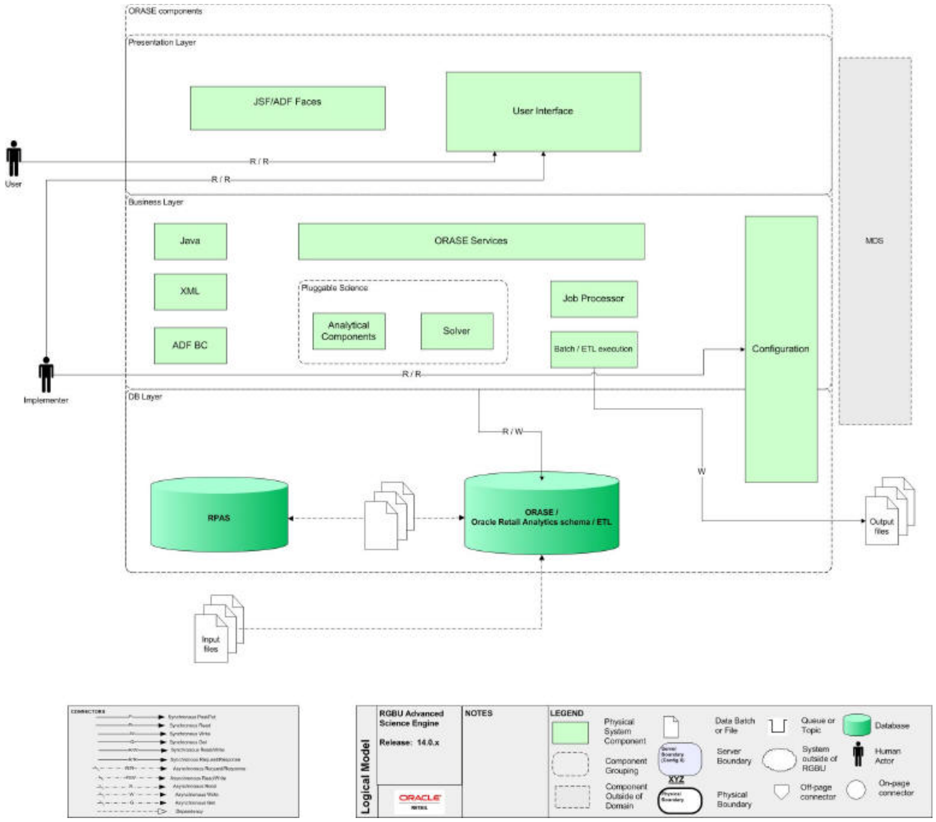
This chapter describes, at a high level, the overall ORASE architecture and some sample deployment options and summarizes the ORASE implementations and installations. It contains the following sections:

- ORASE Architecture Overview
- ORASE Deployment Components
- Hardware and Software Requirements
- ORASE Data Architecture
- ORASE High-Level Data Model
- ASO High-Level Data Model
- Sample Deployments

## ORASE Architecture Overview

The main ORASE components are illustrated in Figure 2–1, "ORASE Component Architecture".

*Figure 2–1 ORASE Component Architecture*



ORASE is a three-tiered architecture consisting of a Presentation Layer, Business Layer, and Database Layer.

The Presentation Layer is implemented using Java Server Faces (JSF) and Oracle Application Development Framework (ADF) Faces.

The Business Layer contains:

- Java code and XML
- ADF Business Components (BC)
- Java Persistence Application Programming Interface (JPA)
- ORASE services that contain common utilities, common logging, capture, and playback (as appropriate)
- Pluggable science that contains calculation algorithms and the Gurobi solver (for ASO only)
- Batch and Extract-Transform-Load (ETL) scripts orchestration
- Job Processor for stage and run execution (Oracle Coherence is optional for scaling)

Metadata is used throughout the Business and Presentation Layers.

The Database layer contains all persisted data and corresponding data mining, common, and module-specific procedures. The major components of the Database layer are:

- RPAS, on which CM and the Fusion Client are installed. ORASE communicates with RPAS only via ETL scripts.

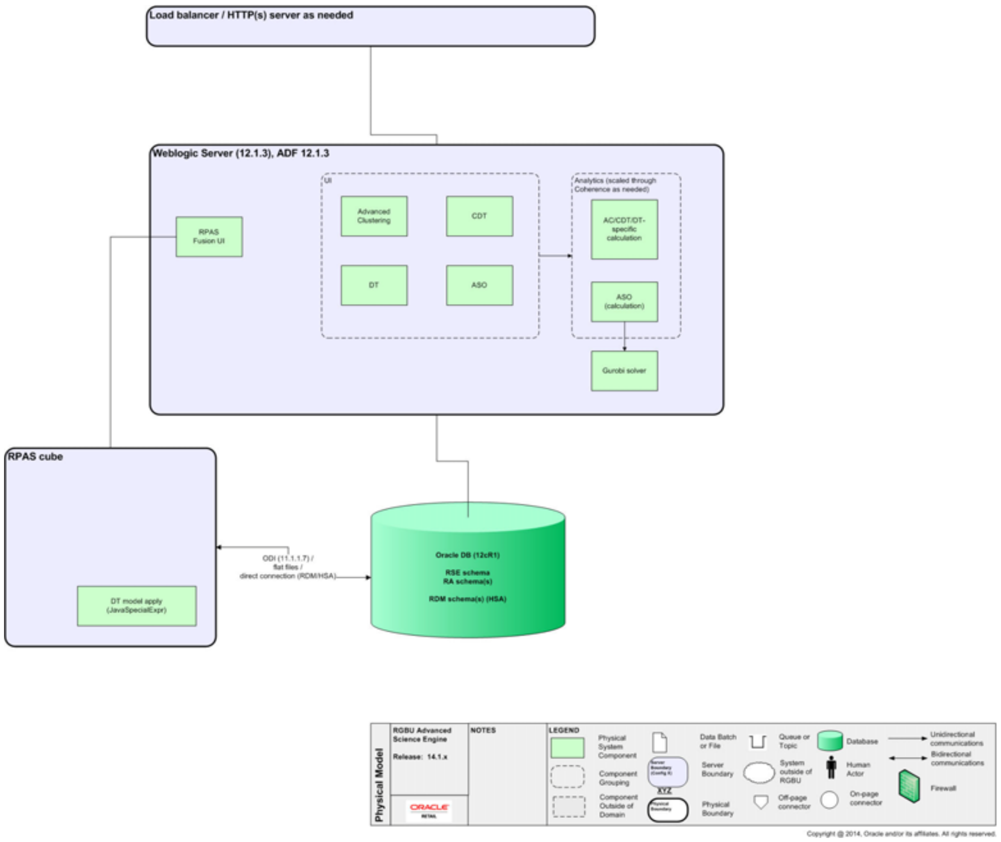
- The ORASE database.
- The RA Data Model.

Configuration is used throughout the business and database layers. See Chapter 5, "Configuration" for details.

## ORASE Deployment Components

Figure 2–2, "ORASE Deployment Components" outlines the deployment components that are included with ORASE. (Note that MBA is used only by RA.)

*Figure 2–2 ORASE Deployment Components*


Load balancer / HTTP(s) server as needed
Weblogic Server (12.1.3), ADF 12.1.3
RPAS Fusion UI
UI
Advanced Clustering
CDT
DT
ASO
Analytics (scaled through Coherence as needed)
AC/CDT/DT-specific calculation
ASO (calculation)
Gurobi solver
RPAS cube
DT model apply (JavaSpecialExpr)
ODI (11.1.1.7) / flat files / direct connection (RDM/HSA)
Oracle DB (12cR1)
RSE schema
RA schema(s)
RDM schema(s) (HSA)
Physical Model
RGBU Advanced Science Engine
Release: 14.1.x
ORACLE RETAIL
NOTES
LEGEND
Physical System Component
Component Grouping
Component Outside of Domain
Server Boundary (Config X)
XYZ
Physical Boundary
Data Batch or File
Server Boundary
Physical Boundary
Queue or Topic
System outside of RGBU
Off-page connector
Database
Human Actor
On-page connector
Unidirectional communications
Bidirectional communications
Firewall
Copyright @ 2014, Oracle and/or its affiliates. All rights reserved.

The Oracle WebLogic Application Server hosts the user interface and middle-tier components.

ORASE is the umbrella for all the related applications.

The Oracle Database hosts the ORASE back-end components and the RA Data Model (RADM), also known as the RA Schema.

Computation components run either on WebLogic or, as shown, optionally on computation nodes using Oracle Coherence.

Data interaction between the Oracle Database and RPAS is done through a file-based interface/ODI.

For security, as needed, the UI and installer supports HTTPS and SSL communication. Oracle wallet is used to store database connection credentials and provide secure connection to the database.

On ASO only, the Gurobi solver is deployed with the computation components to perform space optimization tasks.

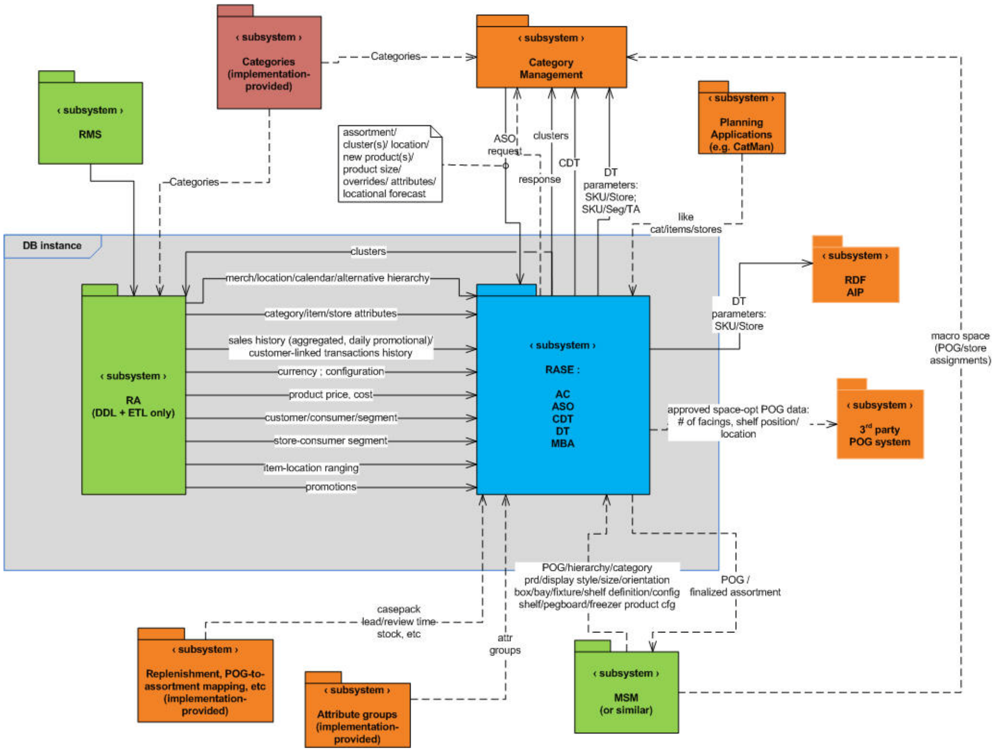
## Hardware and Software Requirements

See the *Oracle Retail Advanced Science Engine Installation Guide* for information about hardware and software requirements.

## ORASE Data Architecture

Figure 2–3, "ORASE Data Architecture" details the overall architecture of RADM, ORASE, and CM/RDF within a single solution.

***Figure 2–3 ORASE Data Architecture***


‹ subsystem ›
Categories (implementation-provided)
Categories
‹ subsystem ›
Category Management
‹ subsystem ›
RMS
‹ subsystem ›
Planning Applications (e.g. CatMan)
assortment/ cluster(s)/ location/ new product(s)/ product size/ overrides/ attributes/ locational forecast
ASO request
clusters
CDT
response
DT parameters: SKU/Store; SKU/Seg/TA
like cat/items/stores
Categories
DB instance
clusters
merch/location/calendar/alternative hierarchy
category/item/store attributes
sales history (aggregated, daily promotional)/ customer-linked transactions history
currency ; configuration
product price, cost
customer/consumer/segment
store-consumer segment
item-location ranging
promotions
‹ subsystem ›
RA (DDL + ETL only)
‹ subsystem ›
RASE :
AC
ASO
CDT
DT
MBA
‹ subsystem ›
RDF
AIP
DT parameters: SKU/Store
macro space (POG/store assignments)
approved space-opt POG data: # of facings, shelf position/ location
‹ subsystem ›
3rd party POG system
POG/hierarchy/category prd/display style/size/orientation box/bay/fixture/shelf definition/config shelf/pegboard/freezer product cfg
POG / finalized assortment
casepack lead/review time stock, etc
attr groups
‹ subsystem ›
Replenishment, POG-to-assortment mapping, etc (implementation-provided)
‹ subsystem ›
Attribute groups (implementation-provided)
‹ subsystem ›
MSM (or similar)

Note that, as mentioned above, ORASE requires and depends on the RADM and ETL components being properly installed, configured, and populated with appropriate data required by ORASE. ORASE does not require that the installation of RA, only RADM.

RADM stores core foundation hierarchies and their attributes, (for example, Item, Location, and Calendar), alternate hierarchies for those foundation hierarchies and their attributes (for example, Category Management Groups (CMG) and Trade Areas), Customer/Consumer Segments and their attributes, and performance history. ORASE applications can use this data for their processes.

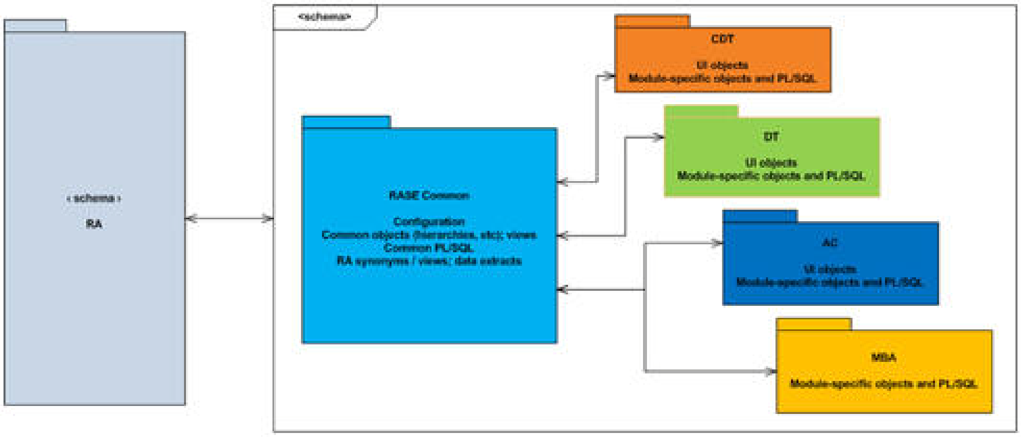
RADM packages ETL code for the Oracle Retail Merchandising System (RMS) foundation hierarchies. In addition to this, RA also has staging tables that can be leveraged for loading alternate hierarchies and their attributes that are not supported in RMS. (See the *Oracle Retail Analytics Operations Guide* for details about the API.) RMS is not considered the system of record for CMG, Trade Area, and

Customer/Consumer Segments; therefore, extracting this information is the system integrator's responsibility. Once this data is loaded in RA staging tables, it is consumed by RA's ODI processes. (See Appendix B, "Database Detail Definitions" for table, column, and mapping definitions.)

## ORASE High-Level Data Model

Figure 2–4, "ORASE Data Model" shows the data flow diagram for the ORASE applications.

***Figure 2–4 ORASE Data Model***


‹ schema ›
RA
<schema>
RASE Common
Configuration
Common objects (hierarchies, etc); views
Common PL/SQL
RA synonyms / views; data extracts
CDT
UI objects
Module-specific objects and PL/SQL
DT
UI objects
Module-specific objects and PL/SQL
AC
UI objects
Module-specific objects and PL/SQL
MBA
Module-specific objects and PL/SQL

As mentioned above, ORASE depends on RADM for the foundation data. Internally, ORASE common includes data objects, configuration, and stored procedures that are common across the ORASE modules.

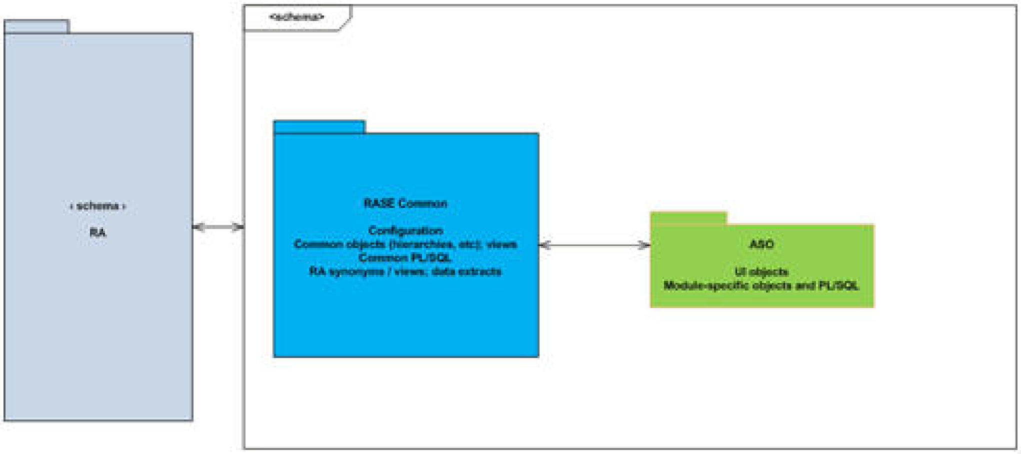
In turn, each ORASE module includes the UI-supporting objects and stored procedures, and the necessary objects and stored procedures, for the data mining and processing activities.

The MBA module uses code in ORASE common and uses RADM for input and output data.

## ASO High-Level Data Model

Figure 2–5, "ASO Data Model"shows the data flow diagram for the ASO application.

***Figure 2–5 ASO Data Model***


‹schema›
RA
<schema>
RASE Common
Configuration
Common objects (hierarchies, etc); views
Common PL/SQL
RA synonyms / views; data extracts
ASO
UI objects
Module-specific objects and PL/SQL

ASO includes the UI-supporting objects and stored procedures and the necessary objects and stored procedures for the optimization and processing activities.

Detailed table definitions can be found in Appendix B, "Database Detail Definitions".

# Sample Deployments

This section describes some sample deployments, including a minimal deployment that can be used for demonstrations and deployments that consider various types of scale, representing real-world customer scenarios. Deployment is illustrated in Figure 2–2, "ORASE Deployment Components".

## Minimal Deployment

It is recommended that you begin the implementation process with a minimal deployment before you begin a complex configuration that involves clustering, computational servers, or RAC. ORASE can be minimally deployed as:

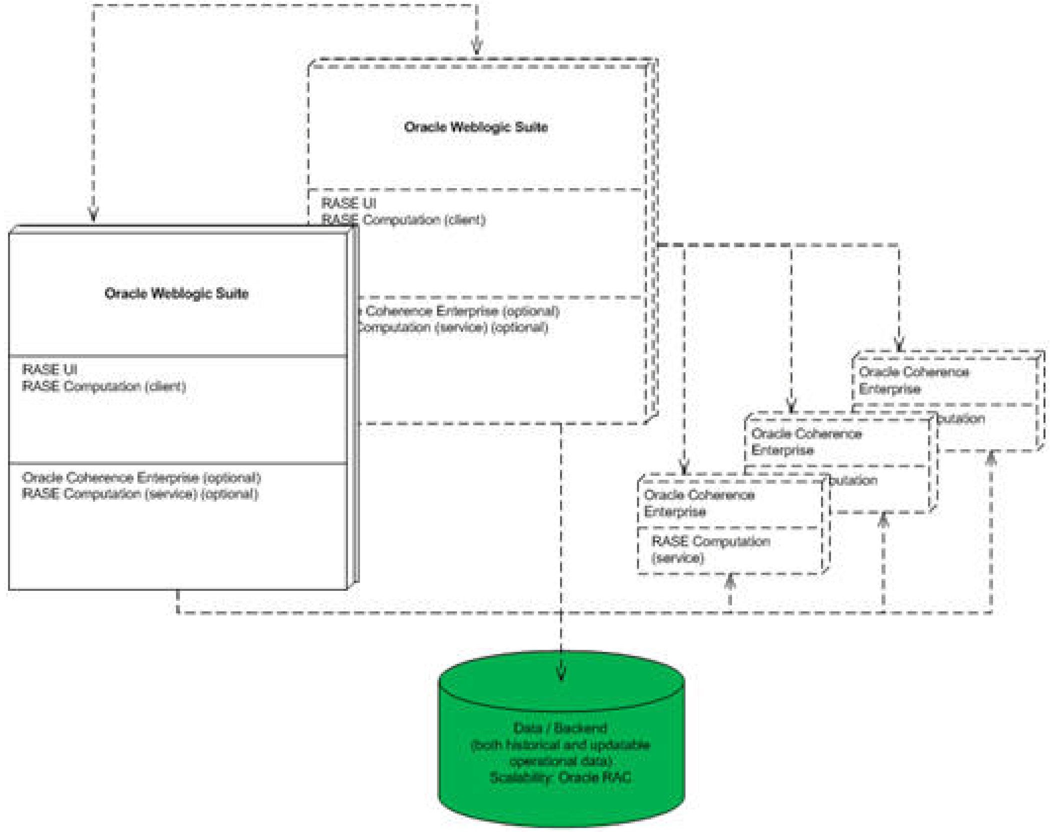
- A single Oracle WebLogic Server instance
- A single Oracle database server

Such deployments are useful for small trials and pilots but do not consider scale and load balancing for users, computation, or database.

## Deployments for Scale

For scaling up to production-level loads, the deployment options shown in Figure 2–6, "Deployment for Scale" can be considered.

*Figure 2–6 Deployment for Scale*


Oracle Weblogic Suite
RASE UI
RASE Computation (client)
Coherence Enterprise (optional)
Computation (service) (optional)
Oracle Weblogic Suite
RASE UI
RASE Computation (client)
Oracle Coherence Enterprise (optional)
RASE Computation (service) (optional)
Oracle Coherence Enterprise
putation
Oracle Coherence Enterprise
putation
Oracle Coherence Enterprise
RASE Computation (service)
Data / Backend
(both historical and updatable operational data)
Scalability: Oracle RAC

Scaling may be done at the UI (Presentation Layer), Computation (Business Layer), or Database layer, as needed.

### User Interface Scaling

To facilitate scaling with multiple simultaneous users, UI/middle-tier scaling can be done using Oracle WebLogic nodes in a cluster. Refer to Oracle WebLogic User documentation for details. Optionally, an external HTTP load balancer can be used.

### Computational Scaling

ORASE, AC, and ASO can make high computational demands on the Business Layer, depending on the amount of data being used, the number of simultaneous users, and the size and quantity of clustering or space optimization runs that are initiated by the user. If computational scale is needed, additional compute nodes can be added using Oracle Coherence to distribute the computational load between these nodes. Note that Coherence can run on the same server where a WebLogic node is running in a separate Java Virtual Machine. See the *Oracle Retail Advanced Science Engine Installation Guide* and the *Oracle Retail Assortment and Space Optimization Installation Guide* for information about Coherence installation.

### Database Scaling

Some ORASE processes, like CDT and DT mining, can make high demands on the database layer, depending on the amount of data being used, the number of users, and the size and number of categories being mined. If database scaling is required, use the

Oracle Real Application Cluster (RAC) solution and add database nodes as needed. Refer to Oracle RAC user documentation for details.

# 3 ORASE Installation and Implementation Overview

This chapter provides a high level step-by-step procedure for the installation and implementation of ORASE applications. Each step refers to more details, either in this document or other documents. It is recommended that you use this chapter as a map, diving into the details for each step and coming back to this to this overview chapter in order to maintain your orientation and to navigate to the next step.

This chapter contains the following sections:

- Prerequisites
- Process Overview
- ORASE Installation and Implementation Process

## Prerequisites

Prior to installing and implementing ORASE, ensure that the following prerequisites have been completed.

The technical stack defined in the Oracle Retail Advanced Science Engine Installation Guide must be installed, configured, and at the correct revision level. The RA Data Model (RADM) must be installed and populated with the data required by ORASE. See Appendix B, "Database Detail Definitions" for details about the required RA tables.

## Process Overview

Here is a high-level overview of the steps involved in the deployment, installation, and implementation of ORASE. The details about customization, attribute pre-processing, configuration, and various options or alternatives are provided later in this chapter in the sections Error: Reference source not found and Error: Reference source not found

It is recommended that you first complete a minimal deployment before starting a complex configuration that involves computational servers, WebLogic clustering, and RAC.

### Platform Implementation Overview

You must complete the following platform deployment steps before you install ORASE. Platform Implementation Steps for the required specifications.

1. Install and configure RPAS and Category Management.

2. Install and configure the Oracle database server(s).
3. Install and configure RADM.
4. Load the dimension data required by the ORASE applications into RADM.
5. Install and configure the application server(s).
6. Install the WebLogic server and ADF on the application server(s).
7. Optionally, install and configure the computation server(s) for AC and/or ASO.

## ORASE Installation and Implementation Overview

Here are the steps to follow to complete the installation and implementation of ORASE. See ORASE Installation and Implementation Steps for details.

1. Install the RPAS Fusion Client (if installing ASO MicroApp)
2. Install the ORASE Database Components (AC, ASO, CDT, DT and/or MBA) as needed.
3. Install the ORASE Application Components (AC, ASO MicroApp, ASO Standalone, CDT, DT, and/or MBA), as needed.
4. Configure the ORASE application roles and users.
5. Edit and load common seed data.
6. Perform attribute preprocessing for CDT and DT, as appropriate.
7. For each of the applications you have installed (AC, ASO, CDT, DT, and/or MBA):
   a. Edit the application's rse_config.ctl seed configuration data file.
   b. If installing ASO Database Component:

      Perform any ASO-specific attribute preprocessing.
   c. Load the seed configuration and application data.
   d. If installing ASO Database Component:

      Update the planogram-to-assortment mapping table
   e. Test the functionality and execution of the application.
   f. Configure, schedule, and execute the batch scripts.

# ORASE Installation and Implementation Process

This section provides additional details about the steps outlined in Process Overview.

## Platform Implementation Steps

Before you install the ORASE applications, you must ensure that the underlying platforms are properly implemented. This includes servers, Oracle database, RADM, WebLogic application servers, RPAS, Category Management application, and, optionally, computational servers.

The main steps are listed here, with reference to specific documentation for the details.

See Chapter 2, "Architecture"for an overview of the ORASE platforms and for the specific supported versions for each platform component.

1. Install and configure Category Management/RPAS.

For more information on this process, see the *Oracle Retail Category Management Installation Guide* and *Oracle Retail Category Management Implementation Guide* as well as the Oracle Retail Predictive Application Server documentation.

2. Install and configure the Oracle database server(s).

   For more information on this process, see the Oracle 12cR1 documentation set.

3. Install and configure RADM.

   For more information on this process, see the *Oracle Retail Analytics Installation Guide*.

4. Load the dimension data required by ORASE onto RADM.

   For more information on this process, see *Oracle Retail Analytics Implementation Guide, Oracle Retail Analytics Data Model*, and Appendix E, "Retail Analytics Interfaces Files."

5. Install and configure the application server(s).

   For more information on supported platforms, see the *Oracle Retail Category Management Installation Guide* and *Oracle Retail Category Management Implementation Guide.* In addition, see vendor-specific documentation as necessary.

6. Install the WebLogic server and ADF on the application server(s).

   For more information on configuring WebLogic, see Chapter 5, "Configuration.". In addition. see the Oracle WebLogic 12c documentation set.

7. Optionally, install the RPAS Fusion Client. ASO is a component of CM and is normally installed as a MicroApplication using the same RPAS Fusion Client on which CM is installed. See the *Oracle Retail Category Management Installation Guide* and *Oracle Retail Category Management Implementation Guide* for details on installing and implementing the RPAS Fusion Client for CM.

8. Optionally, install and configure the computation server(s) for AC or ASO. See Chapter 8, "Server Configuration."

   For more information on supported platforms, see the *Oracle Retail Category Management Installation Guide* and *Oracle Retail Category Management Implementation Guide*. In addition, see vendor-specific documentation as necessary.

## ORASE Installation and Implementation Steps

You must complete all the steps described in Platform Implementation Steps before you begin the steps described in this section. 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.

### Install the ORASE Database Components: AC, ASO, CDT, DT, and/or MBA, as Needed

The ORASE installer installs and configures the artifacts that are required by the AC, ASO, CDT, DT, and/or MBA Database Components. Install each of the applications you need, following the instructions provided in the Oracle Retail Advanced Science Engine Installation Guide.

The installer completes the following:

- It copies onto the file system:

  - The ETL and batch scripts for managing data loading, synchronization, and movement
  - Oracle Coherence (which is optional for scaling AC and/or ASO)
- It creates in the Oracle database:
  - ORASE database objects, including tables and views
  - PL/SQL packages (procedures) and types

### Install the ORASE Application Components: AC, ASO MicroApp, ASO Standalone, CDT, DT, and/or MBA, as Needed

The ORASE installer installs and configures the artifacts that are required by the AC, ASO MicroApp, ASO Standalone, CDT, DT, and/or MBA applications. Install each of the applications you need, following the instructions provided in the *Oracle Retail Advanced Science Engine Installation Guide*.

The installer completes the following:

- It deploys on the Oracle WebLogic Application Server
  - The UI/middle tier for each application
  - The ORASE UI and Business Layer applications for each application, as appropriate

### Configure the ORASE Application Roles and Users

Before any user can log into any ORASE application, you must set up application roles, add users, and assign users to the correct roles. To do this, complete the steps described in Chapter 5, "Configuration."

### Demo Dataset

A demo dataset is available to use to configure the applications and then use the applications. It is installed here:

```
$RSE_HOME/dataset/orase_demo_dataset.tgz
```

To use the demo dataset, you can select the Option to have the Installer unzip and load it, or you can untar it to the $RSE_HOME directory

```
tar xvf $RSE_HOME/dataset/orase_demo_dataset.tgz -C $RSE_HOME/
```

### Data Load Overview

> **Note:** Prior to running any installed .ksh scripts, you must source the RSE Environment Setup file located here: $RSE_HOME/common/scripts/lib/rse.env. To source this file, use the command
>
> . $RSE_HOME/common/scripts/lib/rse.env

During an implementation of any ORASE modules, several steps are required after the completion of the installation process. This section provides some details related to these post-install steps.

The rse_config.ksh and the rse_master.ksh script are located in the $RSE_HOME/common/scripts/bin directory. In addition, similar scripts are located within each of the application directories, for example, $RSE_HOME/cdm/cis/scripts/bin has a cis_config.ksh and a cis_master.ksh script. All of the *config.ksh and *master.ksh are similar in nature, so this section focuses on the rse_config.ksh and rse_master.ksh as examples. However, the concepts apply equally to the application-specific *config.ksh and *master.ksh scripts.

#### Configuration Script (rse_config.ksh)

The rse_config.ksh loads all the relevant *.ctl files contained in the $RSE_HOME/common/data/seed_data directory. Upon successful completion of a load of a file, the script maintains a file called processed.lst in the same seed_data directory. Upon subsequent execution of rse_config.ksh, the contents of processed.lst is checked. If a file was previously loaded, no attempt will be made to reload it. This setup prevents the routine from abnormal termination if a step encounters a problem. If a problem does occur with rse_config.ksh and a file is loaded even though the status does not reflect success, the processed.lst file may be manually edited to prevent the script from trying to process that file on subsequence executions. In a similar manner, if a particular process previously succeeded, but it needs to be reloaded, the processed.lst file can be edited to remove that control file so that subsequent executions will re-run that file.

#### Master Setup Scripts (rse_master.ksh)

The rse_master.ksh script facilitates the execution of all routines needed to get the applications ready for use. Prior to running the script, all data in RADM must already be available and any configuration changes must have already been performed.

The rse_master.ksh script uses command line switches to control the execution of different parts of the script. Run rse_master.ksh -9 to see a list of options that are valid for the script. Two options are present to help run multiple steps. The -A option runs all options, and the -R option resumes with the step that follows the -R option. The script has been written so that the options are treated like switches. If you provide an option after the -A or after the initial option provided to -R, then that particular series of steps will not be executed. The default option for running rse_master.ksh (ie. rse_master.ksh -Act) results in the skipping of steps c and step t.

#### Standard Interface Processing

The standard approach for processing inbound data interfaces consists of populating a staging /interface table (which ends with a name of _STG) with data. After the staging table is populated, the data must be processed so it can be incorporated into the appropriate application tables. If any data validation errors occur while the interface is being loaded, the standard process is for the erroneous data to be populated into a matching table that ends with _BAD instead of _STG. This "bad" table contains an ERROR_ROWID column that identifies the row in the _STG table in which a problem occurred. Additionally, an ERROR_DESCR column describes what was invalid about the row. These columns can help determine what to do to resolve the errors and, if necessary, remove the rows from the _STG table.

### Edit and Load Common ORASE Seed Data

All ORASE 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 ORASE applications. The .ctl files for common seed data are located in the directory: <RSE_HOME>/common/data/seed_data. (RSE_HOME is set by the user during installation.)

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 3–1. The remainder are optional and default to reasonable valuables.

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 CM installation and must match for all installed applications.

*Table 3–1 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 CM Group. Recommendation is for a normal installation with Category Management. | 1 |
| RSE | CMGRP_LEVEL_ID | The hierarchy level ID that contains the level of the product hierarchy where the CM Group level exists (installation configuration). Recommendation is for a normal installation with Category Management. | 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$TIMEZONE_NAMES. | America/New_York |
| RSE | PRIMARY_LANGUAGE_CODE | The name of the language code to use for all RSE data sourced from RA. | EN |
| RSE | RA_FISCAL_CAL_ID | The ID of the calendar to use from RA (since RA supports multiple calendars). | 1240 |
| RSE | TRADE_AREA_HIER_TYPE | The hierarchy ID to use for the Trade Area (installation configuration). | 6 |
| RSE | UI_TZ | The time zone for display. It must match SELECT tzname FROM V$TIMEZONE_NAMES. | America/New_York |

For the complete list, see *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1).

### Load the Common ORASE Seed Data

Load the common ORASE seed configuration and data by executing the following load scripts:

```
cd $RSE_HOME/common/scripts/bin
./rse_config.ksh
./rse_master.ksh - Act
```

See Chapter 6, "Data Integration and Interfaces" for details about additional customizations and data load options.

### 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 ORASE database and happens in RA or flat files generated by the user. Once these tables and files are correct, you can load the resulting attributes in the ORASE schema as part of the data load process, as described in the section For Each Application Installed, Complete the Following Steps.

Here are the basic attribute pre-processing steps:

1. Populate RADM with raw attribute data.
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 Chapter 9, "Attribute Processing".

### For Each Application Installed, Complete the Following Steps

All ORASE applications require and depend on RADM data and ETL components. You must install, configure, and populate RADM with the data required by these applications before you install the applications themselves.

As part of the process of loading data for the applications, common RADM data needed by ORASE applications is loaded into the ORASE database. The steps below load RA dimension, hierarchy, attribute, and other common configuration data shared by ORASE applications. This data is loaded only once by the first application loaded (normally CDT). As part of this load process, each application also loads any application-specific configuration and application data it needs.

You must complete the following series of steps for each of the applications (AC, ASO, CDT, DT, and MBA) that you install.

1. Edit rse_config.ctl seed data
2. Edit the remaining application configuration seed data .ctl files
3. Load the seed configuration and application data in each application using the load scripts
4. Test the functionality and execution of the application
5. Configure, schedule, and execute the batch scripts

The seed data load scripts include a config.ksh script for loading configuration and a master.ksh script for loading data. The master.ksh script invokes a number of scripts that load specific data either from staging, RA, or external files. If you want to edit the master.ksh to change the scripts it calls, see *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1) for details.

#### Edit rse_config.ctl seed data

You must edit and load the rse_config.ctl file for each installed ORASE application's configuration data. Each application has its own version of rse_config.ctl in the following directory locations:

- AC: <RSE_HOME>/cdm/cis/data/seed_data
- ASO: <RSE_HOME>/so/data/seed_data
- CDT: <RSE_HOME>/cdm/cdt/data/seed_data
- DT: <RSE_HOME>/cdm/dt/data/seed_data
- MBA: <RSE_HOME>/cdm/mba/data/seed_data

Review rse_config.ctl in the directory listed for each application and adjust any configurations needed for the environment. Some configurations cannot be changed after the application has been used and must be carefully considered now. These are listed in Table 3–2, Table 3–3, and Table 3–4. The rest of the configurations are optional and default to reasonable valuables.

The four major categories of ORASE parameters are:

- CDT and DT time scale, filter, and priority controls
- CDT and DT UI field value and histogram report defaults
- CDT calculation controls, including tree calculations, pruning, demand and replenishment settings (that is, settings for demand and replenishment models)
- DT default controls for attributes, similarities, and DT calculations

All ORASE applications have configurable parameters in the RSE_CONFIG table. All have default values and are configurable and editable by the administrator. In general, if a user does not select a value for a particular field, it will default to the value listed in this table. Often, the parameter is not selectable from the UI, and this value is used by the application until it is changed in the database. Parameters that must be initialized at setup are listed in Table 3–2, Table 3–3, and Table 3–4.

See *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1) for the complete list of the configurations that are configurable, updateable by the application, and required at the time of initialization.

Table 3–2 contains the mandatory configuration parameters for CDT.

***Table 3–2 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 |

*Table 3–2 (Cont.) Mandatory CDT Configuration Parameters*

| Application | Parameter | Description | Value |
|---|---|---|---|
| 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 |
| CDT | CDT_PROD_HIER_TYPE | The hierarchy ID to use for the CM Group (installation configuration). The recommendation is for a normal installation with CM | 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 3–3 contains the mandatory configuration parameters for DT.

*Table 3–3 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 CM Group. | 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 3–4 contains the mandatory configuration parameters for AC.

*Table 3–4 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.

*Table 3–5 Mandatory ASO Configuration Parameters*

| Application | Parameter | Description | Value |
|---|---|---|---|
| SO | SO_CAL_HIER_TYPE | The hierarchy ID to use for the calendar (installation configuration). | 10 |
| SO | SO_FISCAL_CAL_HIER_TYPE | The hierarchy ID to use for the fiscal calendar (installation configuration). | 11 |
| SO | SO_LOC_HIER_TYPE | The hierarchy ID to use for location (installation configuration). | 2 |
| SO | SO_PROD_HIER_LEVEL_FOR_LEAF_NODE | The product hierarchy level number for leaf node. | 7 |
| SO | SO_PROD_HIER_TYPE | The hierarchy ID to use for the product (installation configuration). | 1 |

For the complete list of ASO configuration parameters, see *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1).

### Prepare the ASO Seed Data for Loading

The following seed data must be loaded for ASO. All of these files are loaded by so_config.ksh from the <RSE_HOME>/so/data/infile directory. You should review the config script and verify the exact name and location of each file that is expected to be loaded.

- ASO application seed data. The ASO seed data .ctl files are in the directory <RSE_HOME> /so/data/seed_data. The default .ctl files control loading input files that match the input file interfaces defined in Chapter 6, "Data Integration and Interfaces". If you have any custom seed data, any .ctl files they need should be placed here and executed manually.
- CM input file data, including assortment data, new items, and forecast. CM input files are part of the standard file interface between CM and ASO, created by the CM application as part of an ASO request. This data is loaded as part of the normal batch process. See the *Oracle Retail Category Management User Guide* and *Oracle Retail Category Management Implementation Guide* for details.
- External file data, including planograms and replenishment. External data is loaded from user-created files as needed via ETL scripts. This includes data on

assortments, product, forecasts, planogram display, and replenishment. You must create these import files according to the file interface definitions defined in Chapter 6, "Data Integration and Interfaces".

### Perform the ASO-Specific Attribute Preprocessing

ASO uses product attributes, but for a different purpose than ORASE. In ASO, product attributes are used to define product placement constraints during space optimization. Because of this, ASO may have some product constraints that differ from CDT or DT product constraints. The resulting attributes are loaded in the ASO schema as part of the ORASE data load process.

Additional background information and the steps to do this are defined under Chapter 9, "Attribute Processing".

### Edit the remaining application configuration seed data .ctl files

Review any additional seed data .ctl files in the directory for each application and adjust any as needed for the specific application.

> **Note:** The merchandise hierarchy levels defined in rse_hier_level.ctl must align with the common data that was loaded from RADM.

### Load the seed configuration and application data in each application using the load scripts

Execute the following load scripts.

For AC:

```
cd $CIS_HOME/scripts/bin
./cis_config.ksh
./cis_master.ksh -A
```

See Chapter 5, "Configuration" for details about the additional AC customization and configuration capability.

For ASO:

```
cd $ASO_HOME/scripts/bin
./so_config.ksh
./so_master.ksh -A
```

For CDT:

```
cd $CDT_HOME/scripts/bin
./cdt_config.ksh
./cdt_master.ksh -A
```

For DT:

```
cd $DT_HOME/scripts/bin
./dt_config.ksh
./dt_master.ksh -A
```

For MBA:

```
cd $MBA_HOME/scripts/bin
./mba_config.ksh
./mba_master.ksh -A
```

See Chapter 7, "Market Basket Analysis Overview" for details about the additional MBA customization and configuration capability.

### Update the Planogram-to-Assortment Mapping

The relationships between assortment and planogram data that you loaded must be defined. This mapping simplifies the job of the application planner by partially automating the mapping of sets of planograms with assortments. This mapping process also matches the seasonality of planograms and assortments and considers demand spread factors for products that are assigned to multiple planogram sets at one time. You do this by creating or editing a pair of mapping files. Batch scripts automate the loading of these files and the subsequent mapping process in order to define the relationships of the planograms to the assortments.

The planogram-assortment mapping files must be created and loaded before either the assortments or the planograms can be used by ASO. If you load any new planograms or assortments into ASO after the initial mapping, you must update and load the new mapping files before the new planograms and assortments can be used in ASO. These files are:

- Assortment and Space Optimization POG to Assortment Mapping File. This file contains the planogram hierarchy to assortment product mapping information. This data is used to identify which planogram should be used for each product.
- Assortment and Space Optimization POG Season to Assortment Mapping File. This file contains the planogram season to assortment date mapping. Once the mapping from product to planogram has been performed, a second pass is used to identify the specific planogram and assortment(s) in the table for the seasonal time period.

The details for these mapping files are defined in Chapter 6, "Data Integration and Interfaces."

#### Test the functionality and execution of the application

Each ORASE application must be tested for basic functionality and any issues with the deployment, installation, implementation be resolved before proceeding. See the *Oracle Retail Advanced Science Engine User Guide* and the *Oracle Retail Assortment and Space Optimization User Guide* for details about using each application.

Note that the WebLogic domain must be restarted in order to load the latest database configuration changes before everything is fully functional because changes to RSE_CONFIG are only picked up at startup.

#### Configure, schedule, and execute the batch scripts

For each application, periodic batch scripts must be scheduled to load ongoing application data and keep the various dimensions up to date and in synchronization. Export processes can also be set up in order to export data to other applications. For the list of batch processes, their order, priority and frequency, see *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1).

### ASO MicroApp Fusion Client Integration

When you are installing ASO into the RPAS Fusion Client as a Micro application, you must configure the task flows so that they appear in CM's Fusion Client UI.

### Module Taskflow Configuration for ASO

Insert the following module_step elements in the file TaskflowMultiSolution.xml. Place them inside the RPAS task element that corresponds to the desired parent task-Flevel node in the RPAS navigation tree. The TaskflowMultiSolution.xml can be found in the <FC install dir>/MultiSolution directory. For an example of the complete file, see Appendix A, "Sample Taskflow_MultiSolution.xml File for ASO/Fusion Client Integration."

```
<module_step>
        <name>sia-so.asomain.microapp.name</name>
        <description>sia-so.asomain.microapp.desc</description>
        <module>asomain</module>
        <module_bundle>sia-so</module_bundle>
        <order_num>6</order_num>
      </module_step>
```

This entry must be added twice by default.

- First instance: under Activity1.Task2.Step8 (this comes under the Assortment Planning at Cluster task). The <order_num> tag here is important, and it should be <order_num>6</order_num>.
- Second instance: under Activity1.Task3.Step10 (this comes under the Assortment Planning at Store task) The <order_num> tag here should be <order_num>2</order_num>.

For the above XML, the sub-elements of the module step are described in Table 3–6.

***Table 3–6 Sub-Elements***

| Element | Description |
|---|---|
| Name | Resource key for the text used for rendering the module link in the navigation tree. The key is required to be defined in resources/MultiSolutionBundle.properties. |
| Description | Resource key for the descriptive text in the pop-up that appears when the user hovers over the module link. The key is to be defined in resources/MultiSolutionBundle.properties. |
| Module | The name of the plug-in (that is, module) that is specified in the bundle manifest file. The bundle manifest file is named <bundle-name>-bundle-manifest.xml and is located in <FC-install-dir>/functionalmodulebundles/<bundle-name>. |
| module_bundle | The name of the plug-in bundle. It can be obtained from the bundle manifest file. See the description of the module element above. |
| order_num | This dictates the position of the module link in relation to all the nodes under the parent RPAS task. |

### Resource Keys Defined in MultiSolutionBundle.properties

As described in Table 3–6, the resource keys used in the module link definitions are defined in the file:

<FC install dir>/MultiSolution/resources/MultiSolutionBundle.properties.

The entries are:

sia-so.asomain.microapp.name=Assortment and Space Optimization

sia-so.asomain.microapp.desc=Assortment and Space Optimization

A sample Taskflow_MultiSolution.xml can be found in Appendix A, "Sample Taskflow_MultiSolution.xml File for ASO/Fusion Client Integration".

## ASO Gurobi Solver Configuration

**Note:** Gurobi is only required for ASO Standalone and ASO MicroApp.

### Enabling Gurobi for ASO

After installation, <RSE_HOME> contains:

```
<RSE_HOME>/so/export/gurobi.jar
```

[the gurobi.jar is platform specific]

<RSE_HOME> also contains one of the following sets of files, depending upon the platform.

- AIX

  ```
  <RSE_HOME>/so/export/aix64/libGurobiJni55.a
  <RSE_HOME>/so/export/aix64/libgurobi55.a
  ```

- Linux

  ```
  <RSE_HOME>/so/export/linux64/libGurobiJni55.so
  <RSE_HOME>/so/export/linux64/libgurobi55.so
  ```

- Solaris

  ```
  <RSE_HOME>/so/export/solaris64/libGurobiJni55.so
  <RSE_HOME>/so/export/solaris64/libgurobi55.so
  ```

### Adding Gurobi JNI Directory to PATCH_LIBPATH for WLS

Edit <WLS_DOMAIN_HOME>/bin/setDomainEnv.sh

After these lines (near Line 209):

```
Path and Path for this domain,
# Please uncomment the following lines and add a valid value for the environment
variables
# set PATCH_CLASSPATH=[myPatchClasspath] (windows)
# set PATCH_LIBPATH=[myPatchLibpath] (windows)
# set PATCH_PATH=[myPatchPath] (windows)
# PATCH_CLASSPATH=[myPatchClasspath] (unix)
# PATCH_LIBPATH=[myPatchLibpath] (unix)
# PATCH_PATH=[myPatchPath] (unix)
```

If installing on AIX, add these two lines after the above.

```
PATCH_LIBPATH=<RSE_HOME>/so/export/aix64
export PATCH_LIBPATH
```

If installing on Linux, add these two lines after the above.

```
PATCH_LIBPATH=<RSE_HOME>/so/export/linux64
export PATCH_LIBPATH
```

If installing on Solaris, add these two lines after the above.

```
PATCH_LIBPATH=<RSE_HOME>/so/export/solaris64
export PATCH_LIBPATH
```

# 4

# ORASE Common Functionality

This chapter addresses functionality that is common throughout ORASE and contains the following sections:

- Service Manager
- Restart and Recovery
- Debugging and Message Logging
- As Is Aggregates
- Maintenance

## Service Manager

Service Manager offers a single framework that provides a consistent way of interacting with a service and of queuing multiple requests for a service. Any service that needs a preprocessing step, a processing step, and a postprocessing step, for up to two layers of tasks, can be handled via this service framework.

### Process Flow

Service Manager works in such a way that the invoking application invokes a package to assist with the scheduling and invocation of a specified MBA service. The service is then executed and the results are stored in the MBA staging tables wherever the service implementation needs to store them.

### Service Implementation

There are several components involved in an MBA Service implementation. They include database tables, packages, and extendable object types. The implementation provides provisions for a calling application to request by registering in the RSE_Srvc_Config table and interact with a service for execution and the retrieval of results. It also provides ways of building an execution queue of service requests, which are invoked by a service manager database package RSE_Srvc_mgr.

The MBA DB Service allows the creation of multiple services that you can interact with in a consistent manner. Additionally, each of these services should be able to use parallel processing of multiple service requests.

Services implementation is a two-step process.

1. Set up the service: Queuing the service request so that it can be executed later.
2. Process the queue: Executing the service queue that was set up.

Setting up a service can be done in the following way:

```
begin
  rse_srvc_mgr.batch_pre_process(<ServiceType>,<ServiceName>);
end;
```

Processing the queue can be done in the following way:

```
begin
  rse_srvc_mgr.process_queue(<ServiceType>);
end;
```

<ServiceType> and <ServiceName> are the valid Service Types and Service Names configured in the Service configuration table RSE_SRVC_CONFIG.

## Service Objects

This section defines the database tables used to implement the Service Manager and the routines used by the Service Manager.

### Service Tables

Table 4–1 defines the database tables being used to implement the service manager feature.

*Table 4–1 Service Manager Database Tables*

| Name | Description |
|---|---|
| RSE_SRVC_TYPE | This table contains a list of service types, to which a specific service implementation can belong. This allows grouping of related service requests. Example service types are ARM and BL. |
| RSE_SRVC_CONFIG | This table contains the service configurations to be used for the different MBA services and is configured as a part of installation. Though this table allows for reconfiguring services in order to implement different service implementation, this is generally not intended for customer configuration. |
| RSE_SRVC_REQ_STATUS | This table defines the status of the different service requests. The contents of this table are implemented as rows in this table, and also as constants with the RSE_SRVC_MGR package. |
| RSE_SRVC_REQ_QUEUE | This table contains a queue of service requests that are to be executed by the Service Management database package. This table allows the storing of large number of requests, so that they can be executed in parallel threads and in a defined order. |
| RSE_SRVC_REQ_QUEUE_PROP | This table contains the various property key and values that are used for the execution of a queued service request. |

### Service Package: RSE_Srvc_Mgr

Service Manager uses a database package RSE_Srvc_Mgr that contains various helper routines, listed in Table 4–2, to assist with the setup, execution, and clean up of services and any service requests.

*Table 4–2 Service Package Routines*

| Name | Input Parameters and Types | Description |
|---|---|---|
| batch_pre_process | p_srvc_type VARCHAR2,<br>p_srvc_name VARCHAR2,<br>p_cancel_failed_srvc_scope VARCHAR2 DEFAULT 'TYPE' | Helper routine to retrieve an MBA Service, set up the service for execution, and add it to the service queue. |
| get_srvc | p_srvc_type VARCHAR2,<br>p_srvc_name VARCHAR2 | Routine to retrieve a service based on the service identification that is provided. |
| process_queue | p_srvc_type VARCHAR2,<br>p_srvc_name VARCHAR2 | Routine to initiate processing of a queue of service requests. |
| cancel_srvc | p_srvc_type VARCHAR2,<br>p_srvc_name VARCHAR2,<br>p_srvc_req_grp VARCHAR2,<br>p_srvc_id NUMBER | Cancels the execution of either an entire service queue, a service request group, or an individual service request. |
| cancel_failed_srvc_req | p_srvc_type VARCHAR2,<br>p_srvc_name VARCHAR2 | Cancels any failed service requests that relate to the provided parameters. |
| pause_srvc | p_srvc_type VARCHAR2,<br>p_srvc_name VARCHAR2,<br>p_srvc_req_grp VARCHAR2,<br>p_srvc_id NUMBER | Pauses the execution of either an entire service queue, a service request group, or an individual service request. |
| resume_srvc | p_srvc_type VARCHAR2,<br>p_srvc_name VARCHAR2,<br>p_srvc_req_grp VARCHAR2,<br>p_srvc_id NUMBER | Resumes the execution of either an entire service queue, a service request group, or an individual service request. |
| retry_srvc | p_srvc_type VARCHAR2,<br>p_srvc_name VARCHAR2 | Retries the execution of any failed service requests within an entire service queue, a service request group, or an individual service request. |

## Restart and Recovery

The services invoke two routines in which one of them initializes the process queue by invoking rse_srvc_mgr.batch_pre_process and the other one executes the process queue by invoking rse_srvc_mgr.process_queue. If the process fails during any of the setup process queue, it may be restarted. There is no adverse effect to re-running the setup process queue that has already failed until the process queue gets executed.

However, if a failure occurs during the process execution stage, once the underlying cause of that failure has been resolved, the execute process may be restarted, and it will restart any failed processes. It is important to note that while resolving a failure in the process execution step, the process queue initialization step should not be rerun. Doing that would leave transient tables, and would end up reprocessing previously completed steps, which may have already been written to the staging tables. This in turn would cause unique constraint errors when those processes are rerun.

## Debugging and Message Logging

Various levels of compilation options are available in the code being installed. They can be enabled or disabled in any desired combination. None of these options must be enabled, but if additional debugging information is required, these settings can help:

- DEBUG_MODE provides various debugging information and logging for a routine.
- DEBUG_DETAIL provides detailed debugging information. This type of information is more verbose.
- DEBUG_DATA allows retention of any transient data objects that a process creates in order to enable a closer review of the data that the process creates.
- DEBUG_TRACE captures the start and end times of most routines, along with the parameters used for the routine, in the log table.

Not all objects support all debugging options, but enabling an option that is not used will produce no adverse effects.

An example of enabling all of the above compilation options is:

```
alter type MBA_ARM_RTL_TOP10_T compile PLSQL_CCFLAGS = 'DEBUG_MODE:TRUE, DEBUG_
DATA:TRUE, DEBUG_DETAIL:TRUE, DEBUG_TRACE:TRUE';
```

An example of disabling all of the above options:

```
alter type MBA_ARM_RTL_TOP10_T compile PLSQL_CCFLAGS = 'DEBUG_MODE:FALSE, DEBUG_
DATA:FALSE, DEBUG_DETAIL:FALSE, DEBUG_TRACE:FALSE';
```

An example of enabling only DEBUG_MODE:

```
alter type MBA_ARM_RTL_TOP10_T compile PLSQL_CCFLAGS = 'DEBUG_MODE:TRUE';
```

## Logging Table

The error messages for errors in the processing and the debug messages due to the enabling one of the compilation option are stored in the table RSE_LOG_MSG. The following information is available in the table.

- Debug or Error Logging Message
- Program Units and Routine Name associated
- Logging Level to identify if the message is due to debug compilation options or errors
- Date and Time when the message was registered

## Transient Tables

Transient tables that are created internally during processing, which are named with the prefix TMP$ internally, are generally deleted once the process gets finished. When the application enables the DEBUG_DATA compilation option to facilitate the capturing of any transient objects, the tables are archived into a DEBUG$ table of a similar name without the identifying prefix of the temporary tables. These tables become partitions within the matching DEBUG$ table, where the partition ID is equal to the service request ID prefixed with 'P_', which created the transient data.

When enabled, DEBUG_DATA retains transient data and occupies space in the database, so it should be used with caution. This option should never be used when running in a weekly production batch schedule.   The option should be enabled only for special runs of the process in order to diagnose any issues with a process and in cases where the issues cannot be found out without retaining the transient data. It should be disabled as soon as the process to be diagnosed is finished. Keeping DEBUG_DATA option enabled will produce adverse DB Sizing effects.

For example, if there is a problem with the MBA's ARM Service ANC_SC and debugging with retaining the data is required to analyze the data, enabling debug on the generic ARM Service, the TOP10 Service, and the ANC_SC Service are needed since all three are involved. Refer to Figure 7–4, "Market Basket Analysis ARM Services Object Types Hierarchy" when enabling DEBUG_DATA for a service so that all levels have consistent support. After running the services, disable debug on these services, perform whatever data review is required to determine the issue, and clean up the temporary tables.

To enable the DEBUG_DATA for ANC_SC Service:

```
alter type MBA_ARM_RTL_ANC_SC_T compile PLSQL_CCFLAGS = 'DEBUG_DATA:TRUE';

alter type MBA_ARM_RTL_TOP10_T compile PLSQL_CCFLAGS = 'DEBUG_DATA:TRUE';

alter type MBA_ARM_SRVC_T compile PLSQL_CCFLAGS = 'DEBUG_DATA:TRUE';
```

To disable DEBUG_DATA for ANC_SC Service:

```
alter type MBA_ARM_RTL_ANC_SC_T compile PLSQL_CCFLAGS = 'DEBUG_DATA:FALSE';

alter type MBA_ARM_RTL_TOP10_T compile PLSQL_CCFLAGS = 'DEBUG_DATA:FALSE';

alter type MBA_ARM_SRVC_T compile PLSQL_CCFLAGS = 'DEBUG_DATA:FALSE';
```

## As Is Aggregates

All of the ARM processing is based on product hierarchy base aggregation results such as Class, Subclass and Department. When the process runs each week, it uses the copies of the product hierarchy as it exists right now.

Baseline service does everything based on as is results. However, as the Baseline processing is at the product level and not hierarchy based, it does not really matter.

## Maintenance

This section contains information about ORASE DB maintenance.

### PROTO$ Tables

ORASE has prototype tables that are created during installation, which are named with the prefix PROTO$. These help to keep only the structure of a database table and in the optimization of the database objects. For example, during processing, MBA reads the PROTO$ tables and dynamically creates permanent or temporary tables exactly like PROTO$ tables structure.

Under normal circumstances, PROTO$ tables do not need to be altered. Other than modifying column properties, any of the table properties such as storage parameters (PCTUSED, PCTFREE, TABLESPACE) or parallel options (PARALLEL) can be modified in the PROTO$ tables if needed to improve MBA processing. Such modifications can impact the temporary or permanent tables that are created based on the PROTO$ tables. Columns should not be modified for any reason. The options that are most suitable for modification include changing the table space, the compression option, and the parallel degree option.

For example, if you change the table PROTO$RSE_CUR_CUST_CUSTSEG_D with PCTUSED=50, PCTFREE=10, TABLESPACE=ETL_DATA_TS with option PARALLEL, the permanent table RSE_CUR_CUST_CUSTSEG_D will be created with

PCTUSED=50, PCTFREE=10 in the table space ETL_DATA_TS with PARALLEL option during ETL processing.

Any PROTO$ tables that have partitions already should not have their partitioning strategy changed. For example, as the tables PROTO$MBA_BL_OUTPUT, PROTO$MBA_BL_RANGE_SLS_A, PROTO$RSE_CUR_CUST_CUSTSEG_D already have partitions, their partition strategy should not be changed.

Any PROTO$ tables that are not already partitioned can be altered to have partitioning if partitioning will improve the database performance. However, it is important to note that if debugging is supported using the DEBUG_DATA compilation option, then the corresponding DEBUG$ table must be altered so that it is sub-partitioned using the same partition strategy in the PROTO$ table.

For example, if you add a hash partition on txn_id in the table PROTO$ARM_WK_SLS_TXN   you should create a hash sub-partition on txn_id in the existing partition, which is based on srvc_req_id on the table DEBUG$ARM_WK_SLS_TXN.

## Service Maintenance Package: RSE_Srvc_Maint

MBA uses the service of the Service Maintenance Package RSE_Srvc_Maint to archive and clean up transient, debug, and log data and ODM Models.

### Archiving Log Data: archive_log_data

This routine helps when moving data partition from regular/online log tables such as RSE_LOG_MSG, RSE_SRVC_REQ_QUEUE and RSE_SRVC_REQ_QUEUE_PROP to historical versions of those same tables. This can reduce the amount of data so that active use of the tables can be more efficient while still allowing retention of the data for historical analysis. Only full or completed weeks and the partitions older than the number of days to preserve (p_days_to_preserve ) is archived to historical table. The current or active week is not considered for archiving.

*Table 4–3    Archiving Log Data*

| Input Parameters | Type | Description |
|---|---|---|
| p_source_table | VARCHAR2 | Name of the log table that requires some partitions archived. |
| p_archieve_table | VARCHAR2 | Name of the historical log table receiving the partitions. Parameter can be null, in which case the default is the first 25 characters from the p_source_table concatenated with string _HIST. |
| p_days_to_preserve | VARCHAR2 | Number of day's worth of log data that should remain in the MBA log table. Defaults to a constant value defined in the package spec using the parameter C_DAYS_TO_PRESERVE. |

Example: To archive the data partitions of the table RSE_LOG_MSG.

```
begin
  rse_srvc_maint.archive_log_data('RSE_LOG_MSG');
end;
```

### Purging Log Data: purge_log_data

This routine helps in cleaning up the old data from the historic log tables by allowing it to retain the desired amount of historic data. Only full or completed weeks and the partitions older than the number of days to preserve (p_days_to_preserve) are purged, and the current or active week is not considered for purging.

*Table 4–4 Purging Log Data*

| Input Parameters | Type | Description |
|---|---|---|
| p_source_table | VARCHAR2 | Name of the table that requires some partitions removed. |
| p_days_to_preserve | VARCHAR2 | Number of day's worth of log data that should remain in the MBA History log table. Defaults to a constant value defined in the package spec using the parameter C_DAYS_TO_PRESERVE. |

Example: To remove old data partitions of the RSE_LOG_MSG_HIST table:

```
begin
  rse_srvc_maint.purge_log_data('RSE_LOG_MSG_HIST');
end;
```

## Archiving MBA Logs: archive_rse_logs

This routine archives old log data partitions from the online tables to the _HIST versions. Only full or completed weeks and the partitions that are older than the number of days to preserve (p_days_to_preserve) are moved.

*Table 4–5 Archiving MBA Logs*

| Input Parameters | Type | Description |
|---|---|---|
| p_hist_tbl_ptrn | VARCHAR2 | This parameter accepts a pattern that is used to identify the historical tables to archive. Default is 'RSE%_HIST'. |
| p_days_to_preserve | VARCHAR2 | Number of day's worth of log data that should remain in the History log table. Defaults to a constant value defined in the package spec using the parameter C_DAYS_TO_PRESERVE. |

Example: To archive old log data partition with the default values.

```
begin
  rse_srvc_maint. archive_rse_logs;
end;
```

## Purging Debug Tables: purge_debug_tables

The purge_debug_tables routine is helpful in removing debug tables once the data analysis is performed to resolve issues due to enabling the compilation option feature debug_data. This routine drops DEBUG partitions corresponding to srvc_req_id older than the number of days to preserve p_days_to_preserve and also eliminates any DEBUG partitions that no longer have a corresponding srvc_req_id.

*Table 4–6 Purging Debug Tables*

| Input Parameters | Type | Description |
|---|---|---|
| p_dbg_tbl_ptrn | VARCHAR2 | This parameter accepts a starting pattern that is used to identify the debug tables to purge. |
| p_days_to_preserve | VARCHAR2 | Number of days worth of debug requests data that should remain in the MBA DEBUG tables. Defaults to a constant value defined in the package spec using the parameter C_DAYS_TO_PRESERVE. |

Example: To purge a debug table with the starting pattern 'ARM'

```
begin
  rse_srvc_maint.purge_debug_tables('ARM');
end;
```

# 5 Configuration

This chapter describes the major configuration points in ORASE, including:

- WebLogic Configuration
- ORASE Database Configuration Details
- Advanced Applications Configuration
- User Interface Configuration
- Advanced Database Customization
- CDT and DT Stage Run Configuration
- CDT, DT, MBA, and AC Configuration Pluggable Science
- Internationalization

> **Note:** Since MBA is distinct from the other ORASE applications, much of what is described here is not applicable for MBA. For clarity, MBA implementation, configuration, operations and data model are described separately in Chapter 7, "Market Basket Analysis Overview.".

## WebLogic Configuration

This section describes details regarding WebLogic configuration.

### Prerequisites

The following are necessary before you configure WebLogic:

- WebLogic Server 12c (12.1.3) must be installed on a supported server.

### User Interface Authentication and Authorization

ORASE uses the Oracle Application Development Framework (ADF) for authentication and authorization. Authentication is done through the Oracle WebLogic-integrated LDAP service.

For authorization, ORASE modules 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 Advanced Science Engine Installation Guide*, *Oracle Retail Advanced Science*

*Engine User Guide, and Oracle Retail Assortment and Space Optimization User Guide* for the definition of standard user roles.

After installing AC, ASO, CDT, and/or DT, a Customer Delegated Administrator uses Oracle Identity Manager create users and place them in appropriate groups.

**Note:** Users cannot log into AC, ASO, CDT, and/or DT until the mapping is done.

This can be done through Oracle Enterprise Manager Fusion Middleware Control that is included with WebLogic Server Standard Edition, WebLogic Server Enterprise Edition, and WebLogic Suite Edition. It is also possible to specify this mapping through Oracle WebLogic scripting. Also, there is an option in the ORASE Installer to have the Installer perform the mapping automatically.

## WebLogic 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. ORASE user management configuration is handled using the WLS Console and the WLS Enterprise Manager (EM).

### User Roles

ORASE supports the roles listed in Table 5–1.

***Table 5–1 ORASE User Roles***

| Role Name | Display Name | Description |
|---|---|---|
| ACAdvancedRole | AC Advanced Analytic User | Responsible for analytical configuration, testing, and cluster analysis in the Advanced Clustering (AC) module. |
| ACBusinessRole | AC Business User | Responsible for analytical configuration, testing, and cluster analysis in the Advanced Clustering (AC) module. |
| CustomerDecisionTreeRole | Maintain Customer Decision Tree | Responsible for analytical configuration, testing, and cluster analysis in the Customer Decision Tree (CDT) module. |
| DemandTransferenceRole | Maintain Demand Transference | Responsible for analytical configuration, testing, and cluster analysis in the Demand Transference (DT) module. |
| MicroSpaceOptAnalyst | Micro Space Optimization Analyst | Responsible for day-to-day Micro ASO activities. |
| CategoryManagement | ASO Category Management User | Product Assortment-centric user who is interested in viewing ASO results and the translation of data between CatManAR and ASO. |
| Administrator | ASO Administrator | Responsible for general system setup and configuration tasks related to Micro ASO |

*Table 5–1 (Cont.) ORASE User Roles*

| Role Name | Display Name | Description |
|---|---|---|
| AnalyticalSuperUser | ASO Analytical Super User | Responsible for analytical configuration, testing, and model diagnosis in MicroASO. |

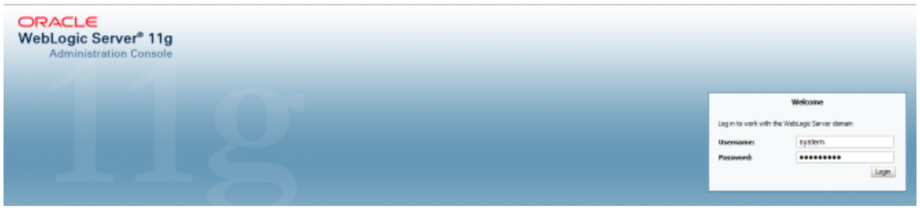
## Configuring Users and Roles

The following steps describe how to set up users and application roles.

### Create Users and Groups

1. Log into your WebLogic Admin console as administration user.

*Figure 5–1 Welcome*



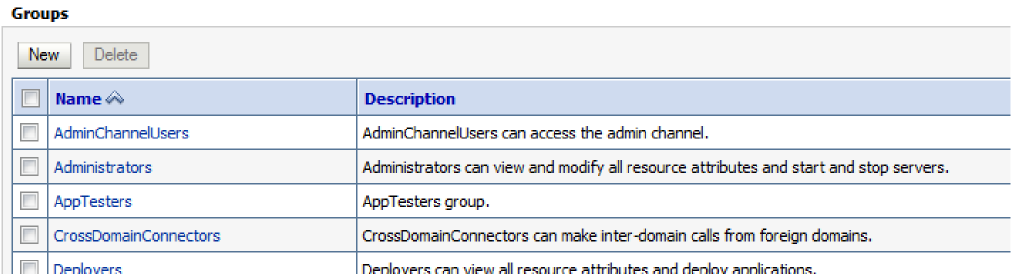
Log in to work with the WebLogic Server domain

Username: system

Password: *********

2. From the left pane, navigate to Security Realms, then select **myrealm** from the Realms list.
3. Select the Users and Groups tab and click the **New** button to create a new user.
4. Enter the desired user information for a new application user and click **OK** to confirm.
5. Navigate to the Groups tab and click the **New** button to create a new group.

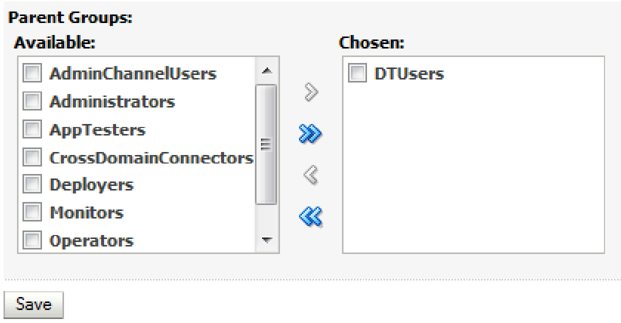
*Figure 5–2 Groups Tab*



| □ | Name | Description |
|---|---|---|
| □ | AdminChannelUsers | AdminChannelUsers can access the admin channel. |
| □ | Administrators | Administrators can view and modify all resource attributes and start and stop servers. |
| □ | AppTesters | AppTesters group. |
| □ | CrossDomainConnectors | CrossDomainConnectors can make inter-domain calls from foreign domains. |
| □ | Deployers | Deployers can view all resource attributes and deploy applications. |

6. Enter a group name (for example, DT_business_users) for the group of users to be assigned to a specific application role (for example, DemandTransferenceRole) and click **OK** to confirm.
7. Select the Users tab again and select the name of the new user. Select the Groups sub-tab and find the new group in the left-side multi-select list. Select the group and click the ">" icon in the center to add the Group to your User.

*Figure 5–3 Parent Groups*



Parent Groups:

Available:

- ☐ AdminChannelUsers
- ☐ Administrators
- ☐ AppTesters
- ☐ CrossDomainConnectors
- ☐ Deployers
- ☐ Monitors
- ☐ Operators

Chosen:

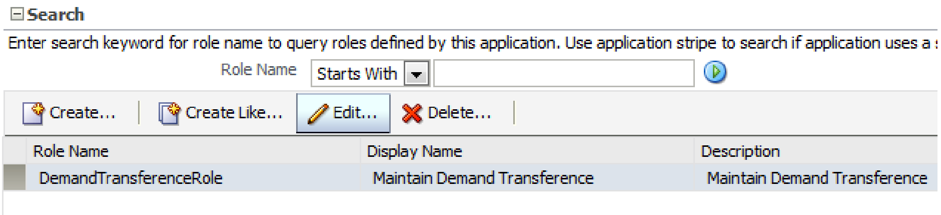
- ☐ DTUsers

8. Click **Save** to persist your changes and log out of your Admin Console.

### Assign Users and Groups to Roles

1. Log into the Enterprise Manager console, locate the application, and right-click to find the Security Application Roles sub-menu.
2. In the Search panel, click the blue and green arrow icon to search for all application roles.

*Figure 5–4 Search Panel*



⊟ Search

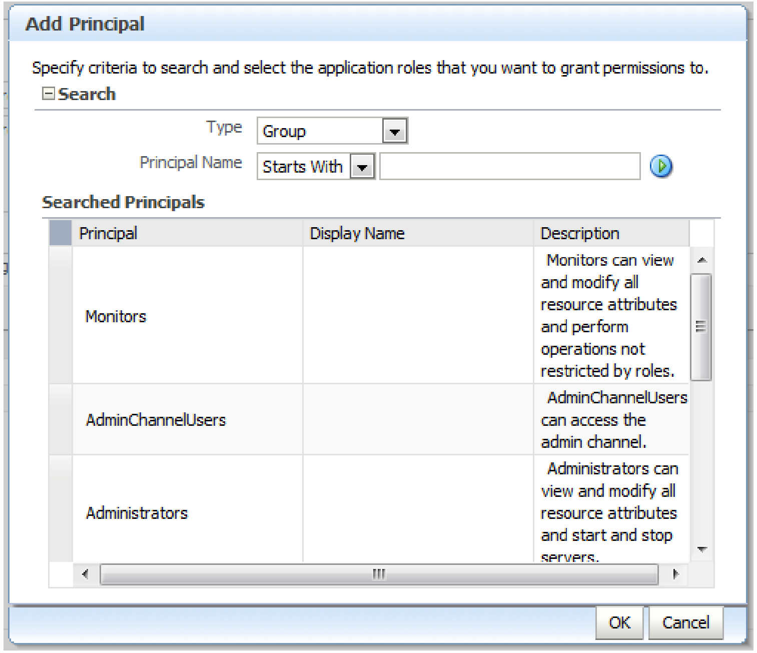
Enter search keyword for role name to query roles defined by this application. Use application stripe to search if application uses a s

Role Name: Starts With

| Role Name | Display Name | Description |
|---|---|---|
| DemandTransferenceRole | Maintain Demand Transference | Maintain Demand Transference |

3. Select the desired application role to add users to and click **Edit**. See User Roles for details.
4. Select **Group** from the Type drop-down menu and then click the blue and green arrow button to search. Select the newly created group name in the list and click **OK**.

*Figure 5–5 Add Principal*



Specify criteria to search and select the application roles that you want to grant permissions to.

Type: Group

Principal Name: Starts With

Searched Principals

| Principal | Display Name | Description |
|---|---|---|
| Monitors | | Monitors can view and modify all resource attributes and perform operations not restricted by roles. |
| AdminChannelUsers | | AdminChannelUsers can access the admin channel. |
| Administrators | | Administrators can view and modify all resource attributes and start and stop servers. |

5. Repeat the previous two steps for any additional users or groups you want to assign to roles. Note that if a user has been assigned to a group, then the user automatically inherits the role when a group is assigned to a role. It is not necessary to have both a group and a user assigned to the same role.
6. When finished, click **OK** in the top right corner of the screen to complete the role mapping.You can now log out of the Enterprise Manager console or continue mapping roles for other applications.
7. You can verify that the users have been successfully created by logging into the appropriate application as the new user.

## WebLogic Clustering (Managed Server Versus Admin Server)

For ASO MicroApp, see the *Oracle Retail Predictive Application Server Administration Guide for the Fusion Client* for information about WLS clustering in the context of a Fusion Client installation.

For AC, ASO standalone, CDT, or DT, WLS Clustering should be installed in a Managed Server setting.

> **Note:** In a Managed Server setting, the MDS Data Source must be targeted to all of the managed servers using it.

## WebLogic (Middle-Tier)-Bound Component Execution Scaling

ORASE components that run WebLogic can be configured to use the available CPU/cores of the node they run on.

Configuration is done in each module's WEB-INF*tojp-application-context.xml where * corresponds to the component. The configuration file itself is stored in the web archive (WAR) file. In RME's case, CDT/DT and AC applications are deployed as shared libraries. Unjar; edit the file, jar and re-deploy; restart WebLogic for changes to take effect.

CDT example:

The fixedThreadPoolSize value specifies the number of threads to spawn. The first bean processes the overall concurrent job requests (default is 2).

The second bean processes CDT step 3, which runs in the middle-tier. In this case, the default number of worker threads is 3 - 2 spawned plus the current one (useCurrentThread is true).

WEB-INF\cdttojp-application-context.xml :

```
<bean id="cdtLocalConcurrentService"
    class="oracle.retail.rse.common.jobprocessor.service.LocalServiceAdapter"
    scope="singleton" lazy-init="true" destroy-method="close">
        <constructor-arg name="logInDb" value="true"/>
        <constructor-arg name="useCurrentThread" value="false"/>
        <constructor-arg name="fixedThreadPoolSize" value="2"/>
</bean>
<bean id="cdtLocalSynchConcurrentServiceNoDbLog"
    class="oracle.retail.rse.common.jobprocessor.service.LocalServiceAdapter"
    scope="singleton" lazy-init="true" destroy-method="close">
        <constructor-arg name="logInDb" value="false"/>
        <constructor-arg name="useCurrentThread" value="true"/>
        <constructor-arg name="fixedThreadPoolSize" value="2"/>
</bean>
```

**Note:** Each module (CDT, DT, AC and ASO) has its own *tojp-application-context.xml files that corresponds to the DB metadata configuration (tables RSE_PROC_TASK_TMPL.CTL and RSE_PROC_TMPL.CTL).

# ORASE Database Configuration Details

The ORASE database includes a set of configurations in the table RSE_CONFIG.

**Note:** Note: ORASE is called RSE in the database. AC is called CIS in the database.

## Common RSE Database Configurations

Table 5–2 describes configuration variables common to all ORASE applications that can be changed by the user. Many of these items requiring initialization cannot be configured later. Most of the listed parameters have default values that are loaded as part of the seed data loading process. However, you should review the default values before you begin to use the application.

See *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1) for the complete list of which configurations are configurable, updateable by the application, and required at initialization.

*Table 5–2 Common RSE Database Configuration Variables*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| RSE | BATCH_ INSERT_SIZE | Batch size for job processor-based processing. | N | 512 |
| RSE | CAL_HIER_ TYPE | The ID of the calendar type that contains the primary calendar. | N | 10 |
| RSE | CAL_PERIOD_ LEVEL | This is the calendar hierarchy level that is used to drive RSE processing. | Y | 4 |
| RSE | CAT_MAN_ APP_ID | The application identifier to use for obtaining external IDs when communicating with CM. | N | 1 |
| RSE | CDM_helpInfo | Either a full URL or a partial URI (after the port) of the static help location. Note that if SSL is configured, full URL must be used. | N | /cdm-help/help/state?navSetId=cdm-help&navId=0 |
| RSE | CHAIN_LEVEL_ DESC | The description to use for any top level hierarchy element when one must be manually created. | Y | CHAIN |
| RSE | CMGRP_HIER_ TYPE | The hierarchy ID to use for the CM Group (installation configuration). | Y | 5 |
| RSE | CMGRP_LEVEL_ ID | The hierarchy level ID that contains the level of the product hierarchy where the CM Group level exists. (Installation configuration). | Y | 2 |
| RSE | CUSTSEG_HIER_ TYPE | The ID of the hierarchy type that contains the customer segment hierarchy. | N | 4 |
| RSE | DEFAULT_APP_ NAME | The default application that is invoked when RSE is run with multiple applications after login. Valid options are:<br>set PARAM_VALUE = 'CDT'<br>set PARAM_VALUE = 'CIS'<br>set PARAM_VALUE = 'DT'<br>set PARAM_VALUE = 'SO' | Y | DT |
| RSE | DEFAULT_SRG_ DOP | The default configuration for any module that does not specify a degree of parallelism. | N | 1 |
| RSE | DEFAULT_SR_ DOP | The default configuration for any module that does not specify a degree of parallelism. | N | 1 |
| RSE | DISPLAY_DATE_ FORMAT | The default date mask to be used by the UI. | N | Mon dd, yyyy |
| RSE | FISCAL_CAL_ HIER_TYPE | The ID of the hierarchy type that contains the fiscal calendar. | N | 11 |
| RSE | LOC_HIER_ TYPE | The ID of the hierarchy type that contains the primary location hierarchy. | N | 2 |
| RSE | MT_TZ | Time zone that is used by application server(s), that is, by the middle-tier. Must match SELECT tzname FROM V$TIMEZONE_NAMES. | Y | America/New_ York |

*Table 5–2 (Cont.) Common RSE Database Configuration Variables*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| RSE | PRIMARY_ LANGUAGE_ CODE | The name of the language code to use for all RSE data sourced from RA. | Y | EN |
| RSE | PROD_HIER_ TYPE | The ID of the hierarchy type that contains the primary product hierarchy. | N | 1 |
| RSE | RA_APP_ID | The application identifier to use for obtaining external IDs when communicating with RA. This should align with RSE_APP_SOURCE. | N | 1 |
| RSE | RA_FISCAL_ CAL_ID | The ID of the calendar to use from RA since RA supports multiple calendars. | Y | 1240 |
| RSE | RDF_APP_ID | The application identifier to use for obtaining external IDs when communicating with RDF. | N | 1 |
| RSE | RSE_SLS_TXN_ SR_DOP | The number of weeks that can be simultaneously processed by the SLS_ TXN_ETL. | N | 1 |
| RSE | ASO_helpInfo | Either a full URL or a partial URI (after the port) of the static help location. Note that if SSL is configured, full URL must be used. | N | /aso-help/help/state?navSetId=aso-help&navId=0 |
| RSE | TRADE_AREA_ HIER_TYPE | The hierarchy ID to use for the trade area (installation configuration). | Y | 6 |
| RSE | UI_TZ | Time zone for display. Must match SELECT tzname FROM V$TIMEZONE_NAMES. | Y | America/New_ York |

# Advanced Applications Configuration

This section describes the advanced configuration available for or required by the ORASE applications.

- CDT and DT configurations
- AC Configuration
- ASO configuration, which includes Gurobi configuration.

## CDT and DT Configuration

This section describes the configuration and parameter settings for CDT and DT.

All ORASE applications have configurable parameters in the RSE_CONFIG table. All have default values and are configurable and editable by the administrator. In general, if the user does not select a value for a particular field, it will default to the value set in this table. Often the parameter is not selectable from the UI, and this value is used by the application until it is changed in the database. Note that some parameters must be initialized at setup. See Table 5–3. Many of these items that require initialization cannot be configured later. See *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1) for the complete list of which configurations are configurable, updateable by the application, and required at initialization.

The following are the major categories of CDT and DT parameters:

- CDT and DT time scale, filter, and priority controls
- CDT and DT UI field value and histogram report defaults
- CDT calculation controls, including trees calculation, pruning, demand, and replenishment settings (the settings for demand and replenishment models)
- DT default controls for attributes, similarities, and DT calculation

*Table 5–3 URLs*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| CDT | START_URL | CDT starting URL | N | /faces/oracle/retail/rse/cdm/fe/view/page/index.jspx |
| CDT | END_URL | CDT logout URL | N | /faces/oracle/retail/rse/cdm/fe/view/page/login.jspx |
| DT | START_URL | DT starting URL | N | /faces/oracle/retail/rse/cdm/fe/view/page/index.jspx |
| DT | END_URL | DT logout URL | N | /faces/oracle/retail/rse/cdm/fe/view/page/login.jspx |

*Table 5–4 CDT and DT*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| CDT | CDT_CAL_HIER_TYPE | The hierarchy ID to use for the fiscal calendar. (Installation configuration). | Y | 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 Week - Installation configuration.) | Y | 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.) | Y | 5 |
| CDT | CDT_CUSTSEG_HIER_TYPE | The hierarchy ID to use for customer segment. (Installation configuration.) | Y | 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.) | Y | 2 |
| CDT | CDT_ESC_PRIORITY | The default priority for escalation. Choices are LOC (location/trade area (default)) or CUST (customer segment). | N | LOC |
| CDT | CDT_EXPORT_START_DATETIME | Start date and time of CDT export. Format: yyyy-mm-dd hh24:mi:ss | N | 11/10/2013 12:11 |
| CDT | CDT_FILTER_SLS_TXN_SR_DOP | The number of parallel threads to use for processing data filtering steps. | N | 1 |

*Table 5–4 (Cont.) CDT and DT*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| CDT | CDT_LOC_ HIER_TYPE | The hierarchy ID to use for location. (Installation configuration). | Y | 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.) | Y | 4 |
| CDT | CDT_PROD_ HIER_TYPE | The hierarchy ID to use for the CM Group. (Installation configuration.) | Y | 1 |
| CDT | CDT_SLS_TXN_ SR_DOP | The number of weeks that can be simultaneously processed by the CDT SLS Txn ETL. | N | 1 |
| CDT | CDT_STATS_ CALC_METHOD | The approach to use for calculating statistics on transient tables (CALC, NONE, or 0-10 for a fixed dynamic sampling level). | N | 2 |
| CDT | CDT_UI_DEF_ CALC_PARENT_ SEGMENT_FLG | UI default for the calculate-only parent customer segments flag. | N | Y |
| CDT | CDT_UI_DEF_ CALC_PARENT_ TRADE_AREA_ FLG | UI default for calculate-only parent trade areas flag. | N | N |
| CDT | CDT_UI_DEF_ CDT_SCORE_ HIST_CNT | UI default for the number of histogram buckets for the CDT scores histogram. | N | 20 |
| CDT | CDT_UI_DEF_ DATA_FILTER_ HIST_CNT | UI default for the number of histogram buckets for the data filtering histograms. | N | 20 |
| CDT | CDT_UI_DEF_ EXCLUDE_ CUST_CNT | UI default for minimum required customer counts for pruning process. | N | 1000 |
| CDT | CDT_UI_DEF_ EXCLUDE_MIN_ SCORE | UI default for minimum CDT score required for the pruning process. | N | 0.25 |
| CDT | CDT_UI_DEF_ EXCLUDE_SKU_ CNT | UI default for minimum number of SKUs for the pruning process. | N | 10 |
| CDT | CDT_UI_DEF_ EXCLUDE_ TREE_LEVEL_ CNT | UI default for minimum number of levels of the tree for the pruning process. | N | 2 |
| CDT | CDT_UI_DEF_ LOWEST_ EXPANSION_ LEVEL | UI default for lowest number of levels allowed for a tree. | N | 15 |
| CDT | CDT_UI_DEF_ MAX_CUST_ AVG_DY_TXN | UI default for maximum number of times greater than average that a customer's daily transaction count can be. | N | 100 |
| CDT | CDT_UI_DEF_ MAX_MISS_ ATTR_CNT | UI default for maximum number of missing attributes a SKU can have. | N | 3 |

*Table 5–4 (Cont.) CDT and DT*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| CDT | CDT_UI_DEF_MIN_ATTR_SKU_CNT | UI default for minimum number of SKUs assigned to an attribute, to be used by the process. | N | 5 |
| CDT | CDT_UI_DEF_MIN_ATTR_VALUE_SKU_CNT | UI default for minimum number of SKUs assigned to an attribute value, to be used by the process. | N | 5 |
| CDT | CDT_UI_DEF_MIN_CUST_TXN_CNT | UI default for minimum number of transactions required for a customer, as a percent of the average number. | N | 0.01 |
| CDT | CDT_UI_DEF_MIN_NODE_ITEM_CNT_PCT | UI default for the minimum percent of SKUs required for a node of the tree before it is considered a terminal node. | N | 0.05 |
| CDT | CDT_UI_DEF_MIN_SKU_TXN_CNT | UI default for minimum number of transactions required for a SKU, as a percent of the average number. | N | 0.01 |
| CDT | CDT_UI_DEF_PRUNING_HIST_CNT | UI default for the number of histogram buckets for the pruning histograms. | N | 20 |
| CDT | CDT_XML_PRECISION | Default precision of weight field in CDT XML. | N | 4 |
| 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. | Y | 15 |
| CDT | HISTOGRAM_DEFAULT_BIN_APPROACH | The default histogram bin approach (C=Custom, W=Width) | N | C |
| CDT | HISTOGRAM_DEFAULT_NUM_BINS | The default number of bins to display for a CDT histogram. | N | 7 |
| CDT | MAX_NUM_WEEKS_FOR_SIMILARITY | 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. | N | 104 |
| CDT | TREE_CALC_ALPHA_LEVEL | The default alpha level for CDT tree calculation. Available options are 1, 2, and 3. | N | 2 |
| CDT | TREE_CALC_METH_OVR | Override for the default approach for calculating the CDT. Available options are PEARSON and SPEARMAN. | N | PEARSON |
| DT | AE_CALC_INT_LENGTH | The number of weeks to group together for in an interval for the AE calculation. | Y | 4 |
| DT | AE_CALC_LOC_JOIN_METHOD | An identifier to define how the AE calculation should limit its joins of the location hierarchy. SELF=Limit data to same location. TA=Limit joins to locations in same Trade Area. FULL=Complete cross product of all locations. | N | SELF |

*Table 5–4 (Cont.) CDT and DT*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| DT | ATTRIBUTE_ LIST_ SEPARATOR | A separator to be used to display a list of attributes in the Similarity Calculation screen. | N | |
| DT | CDT_ SIMILARITY_ AVAILABLE | Whether CDT similarity has been made available to DT. | N | Y |
| DT | DT_BASELINE_ SR_DOP | The number of degrees of parallelism the baseline calculation should use. | N | 1 |
| DT | DT_CAL_HIER_ TYPE | The hierarchy ID to use for the fiscal calendar. | Y | 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.) | Y | 4 |
| DT | DT_CALC_AE_ SR_DOP | The number of degrees of parallelism the DT AE calculation service should use. | N | 1 |
| DT | DT_CAT_LOC_ SKU_CNT_DOP | The number of degrees of parallelism the Category/Location/Week SKU count routine should use. | N | 1 |
| DT | DT_CMGRP_ LEVEL_ID | The hierarchy level ID that contains the level of the product hierarchy that DTs are created for. | Y | 5 |
| DT | DT_CUSTSEG_ HIER_TYPE | The hierarchy ID to use for customer segment. | N | 4 |
| DT | DT_CUSTSEG_ LEVEL_ID | The hierarchy level ID that contains the level of the customer segment hierarchy that DTs are created for. | N | 2 |
| DT | DT_ESC_ PRIORITY | The default priority for escalation. Choices are LOC (location/trade area (default)) or CUST (customer segment). | N | LOC |
| DT | DT_FILTER_SR_ DOP | The number of simultaneous threads to use while filtering data for DT. | N | 1 |
| DT | DT_LOC_HIER_ TYPE | The hierarchy ID to use for location. | Y | 2 |
| DT | DT_LOC_ LEVEL_ID | The hierarchy level ID that contains the level of the location hierarchy that DTs are created for. | Y | 4 |
| DT | DT_MDL_AP_ EXP_WKS_ BACK_END | 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. | N | 1 |
| DT | DT_MDL_MIN_ REQ_LC_SLS_ WKS | The minimum number of weeks per location required for model build. | N | 10 |
| DT | DT_MDL_MIN_ REQ_PR_LC_ SLS_WKS | The minimum number of weeks per product/location required for model build. | N | 10 |
| DT | DT_PROD_ HIER_TYPE | The hierarchy ID to use for the CM Group. | Y | 1 |

*Table 5–4 (Cont.) CDT and DT*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| DT | DT_REMOVE_ REDUNDANCY | If set to Y, then remove redundancy while calculating attribute-based similarities. | N | N |
| DT | DT_SIM_ DISPLAY_ ROWNUM | The number of distinct similarity values to show in the UI pop-up. Setting too high a number effectively eliminates this limit. | N | 9999999 |
| DT | DT_STATS_ CALC_METHOD | The approach to use for calculating statisitcs on transient tables (CALC, NONE, or 0-10 for a fixed dynamic sampling level). | N | 2 |
| DT | DT_WGT_ CALC_SR_DOP | The number of degrees of parallelism the weight calculation script should use. | N | 1 |
| DT | GENERIC_ SEPARATOR | A separator to be used to display a list of items, for example, SKU prod_ext_code name. | N | |
| DT | HISTOGRAM_ DEFAULT_BIN_ APPROACH | The default histogram bin approach (C=Custom, W=Width). | N | W |
| DT | HISTOGRAM_ DEFAULT_ NUM_BINS | The default number of buckets used for the contextual BIs. | N | 7 |
| DT | MAX_NUM_ WEEKS_FOR_ ATTR_WGT | The maximum number of weeks of input data to use for calculating attribute weights. | N | 104 |
| DT | MAX_NUM_ WEEKS_FOR_ AVG_SLS | The maximum number of weeks of input data to use for calculating the average sales. | N | 104 |
| DT | MAX_NUM_ WEEKS_FOR_ FILTERING | 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 will override the other configurations. | N | 104 |
| DT | MAX_NUM_ WEEKS_FOR_ MDL_CALC | The maximum number of weeks that should be used during the model build calculation. | N | 104 |
| DT | MAX_NUM_ WEEKS_FOR_ MDL_UPDT | The maximum number of weeks that should be used during the model build update calculation. | N | 104 |
| DT | MAX_NUM_ WEEKS_FOR_ SIMILARITY | The maximum number of weeks of input data to use for calculating similarity. | N | 104 |
| DT | MAX_SIGNIF_ SIM_CODE | Significant SKUs count is done from this code and up (see SIM_DISPLAY_CODE_ PCT). | N | 3 |
| DT | MDL_AP_EXP_ WKS_BACK_ START | 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. | N | 4 |
| DT | MDL_CALC_ UPDT_MDL_ DOP | The number of parallel threads that can be used to update individual models in a version during the auto update process. | N | 1 |

*Table 5–4 (Cont.) CDT and DT*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| DT | MDL_CALC_UPDT_SCHED_LIMIT | The percentage value (expressed as a decimal) of versions to be attempted to be updated each execution. | N | 0.25 |
| DT | MDL_CALC_UPDT_VERSION_DOP | The number of parallel threads that can be used to update version MDL AE values during the auto update process. | N | 1 |
| DT | POLLING_INTERVAL | Stage status polling interval (in seconds). | N | 30 |
| DT | PRUNED_CATEGORIES_SEPARATOR | A separator to be used to display a list of pruned attributes in the Calculation screen. | N | |
| DT | PR_LOC_STATUS_CHG_WINDOW_LEN | The window length to use for product/location status changes. | N | 1 |
| DT | PR_LOC_STATUS_LAST_COMPLETED_WK | The last completed week for SKU/Store ranging data copying. | Y | 1104005807 |
| DT | SIM_DISPLAY_CODE_PCT | This configuration supports different codes, along with different percent limits for those codes. | N | 0 |
| DT | UI_DEF_CALC_PARENT_CS_ONLY_FLG | The UI default for the calculate only parent customer segments flag. | N | N |
| DT | UI_DEF_CALC_PARENT_TA_ONLY_FLG | The UI default for calculate-only parent trade areas flag. | N | N |
| DT | UI_DEF_MAX_MISS_ATTR_CNT | The maximum number of missing attributes to use for a SKU before requiring it to be filtered from use. | N | 3 |
| DT | UI_DEF_MIN_SKU_CNT | The UI default for minimum number of SKUs required for a segment/store. | N | 10 |
| DT | UI_DEF_MIN_SKU_TXN_LEN_PCT | The UI default for minimum SKU transaction length as a percent of the CM Group average. | N | 0.01 |
| DT | UI_DEF_MIN_TOT_SLS_UNIT_PCT | The UI default for minimum total sales units as a percent of the CM group average. | N | 0.01 |
| DT | WGT_CALC_INTERVAL_LENGTH | The number of weeks to group into an interval that is then used to perform weight calculations. | N | 4 |

## AC Configuration

This section describes the configuration and parameter settings for AC. AC can be installed individually and can be extensively configured using tables specific to AC.

As with CDT, DT, and ASO, AC shares the Common RSE Database Configurations for general configuration. It also has a number of additional tables, described in detail below, for additional configurability.

AC can be configured by:

- Objective (currently only Store is supported),
- Business objective (for example, Category Management), or
- Type criteria (for example, (Consumer Profile/Store Attribute/Product Profile/Product Performance/Mixed Attribute)).

This section describes configurations for the following settings:

- Basic or frequently used settings. These settings are applicable for Business Objective/Objective/Criteria.
- Generic settings. These settings are applicable to the entire AC application. These settings are application-wide settings with pre-configured default values that should be updated at deployment.
- Advanced analytic settings. These settings relate to the BaNG algorithm and are updated by advanced analytic users.
- Rules-based settings. These settings allow extension of rules that are defined for outliers or new stores.

## Basic or Frequently Used Configuration Settings

These configurations listed in Table 5–5 are the most frequently used during implementation and they are pre-populated as part of the application installation using the cis_master.ksh -a Attribute Maintenance option. Further updates to the clustering attributes can be performed using below documentation.

*Table 5–5 Basic or Frequently Used Configuration Settings*

| Name | Configuration Description | Table |
|---|---|---|
| Flexibility to add and remove attributes in the application | Add attributes<br>Remove attributes<br>These attributes are a collection of attributes and can be applied to any criteria/objective/business objective.<br>See How to Insert, Update, or Disable a Field | CIS_TCRITERIA_ATTR |
| Enable criteria for an objective (for example, stores), business objective (for example, Category Management | Enable criteria for a objective/business objective by setting current_flg as on/off.<br>This table also holds settings that are specific to a criteria.<br>See How to Insert, Update, or Disable a Field | CIS_BUS_OBJ_TCRITERIA_XREF |

*Table 5–5 (Cont.) Basic or Frequently Used Configuration Settings*

| Name | Configuration Description | Table |
|---|---|---|
| Enable attributes for a objective (for example, stores), business objective (for example, Category Management), type criteria (for example, consumer profile/store attribute) and attributes applicable to them | Enable attributes for a objective/type criteria/criteria by setting participant flag to 'Y'.<br>Disable attributes for a objective/type criteria/criteria by setting participant flag to 'N.'<br>These attributes are a collection of attributes that are displayed in BI as KPI or attributes that are used for clustering process.<br>See How to Insert, Update, or Disable a Field | CIS_BUS_OBJ_TCRITERIA_ATT_XREF |
| Flexibility to enable or disable nested clustering for a specific criteria | This configuration enables nested criteria under another criterion for each objective/business objective.<br>Example: Consumer Segment can have Store Attribute or Store Attribute can have Store Attribute.<br>See How to Insert, Update, or Disable a Field | CIS_BUS_OBJ_NESTED_TCRITERIA |
| Flexibility to load predefined planning periods into Clustering Solution | Buying or planning periods can be loaded into the system using sqlldr.<br>See How to Insert, Update, or Disable a Field | CIS_EFFECTIVE_PERIOD |

## Generic Configurations

The settings listed in Table 5–6 are applicable only to AC and are generic settings that allows user to format and control AC. Note that the actual application code in RSE_CONFIG uses the acronym CIS instead of the acronym AC. For a complete list of abbreviations, see "Glossary of Acronyms".

Some parameters must be initialized at setup. See Table 5–6. Many of these items that require initialization cannot be configured later. See *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1) for the complete list of which configurations are configurable, updateable by the application, and required at initialization.

*Table 5–6 Advanced Clustering*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| CIS | START_URL | CIS starting URL | N | /faces/oracle/retail/rse/cdm/fe/view/page/index.jspx |
| CIS | END_URL | CIS logout URL | N | /faces/oracle/retail/rse/cdm/fe/view/page/login.jspx |
| CIS | ATTR_NAME_SEPARATOR | The separator character(s) to use to separate the different components of the attribute names in CIS_TCRITERIA_ATTR. | N | |
| CIS | C_ALG_CLUSTER_CATEGORY | Constant declaration for algorithm cluster category. | N | 4 |
| CIS | C_ALG_SPEC_PROPS | Constant declaration for cluster-specific properties | N | 4 |
| CIS | C_ATTR_SCALE_PROPS | Constant declaration for retrieve attribute scale properties. | N | 3 |
| CIS | C_PERCENT_ATTR_TYPE_ID | Constant declaration for percentage clustering attributes type. | N | 2 |
| CIS | C_STMT_CLUSTER_PROPS | Constant declaration for retrieve cluster-specific properties. | N | 1 |
| CIS | C_STMT_SCALE_PROPS | Constant declaration for retrieve scale-specific properties | N | 2 |
| CIS | CIS_DFLT_CALENDAR_HIER_TYPE_ID | Default calendar hierarchy for clustering. | Y | 11 |
| CIS | CIS_DFLT_LOCATION_HIER_TYPE_ID | Default location hierarchy for clustering. | Y | 6 |
| CIS | CIS_DFLT_PRODUCT_HIER_TYPE_ID | Default product hierarchy for clustering. | Y | 5 |
| CIS | CIS_DFT_PIVOT_LVL | Default pivot level shown in explore data. | N | 3 |
| CIS | CIS_NUMERIC_DFT | Default value for numeric attributes shown in BI or UI when values are not available. | N | 0 |
| CIS | CIS_STRING_DFT | Default attribute value for string. | N | UNKNOWN |
| CIS | DEFAULT_APPL_USER_ID | Default user ID for batch processing records. | N | 1 |
| CIS | DEFAULT_CUST_ADVANCED_USER | Identifies the advanced user. | N | AdvancedUser |
| CIS | DEFAULT_CUST_BUSINESS_USER | Identifies the business user. | N | BusinessUser |

*Table 5–6 (Cont.) Advanced Clustering*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| CIS | DEFAULT_ DYNAMIC_ SAMPLING_ LEVEL | Default level of dynamic sampling to apply to sessions | N | 4 |
| CIS | DEFAULT_ NUM_ATTR_ VALUE | Constant for number of discrete values allowed for store attr clustering | N | 15 |
| CIS | DEFAULT_ NUM_BIN | Default number of bins | N | 3 |
| CIS | DEFAULT_STR_ CATEGORICAL_ ATTR | Default String description for row added in cis_tcriteria_attr_type_value table for unmatched grouping | N | UNKNOWN |
| CIS | DEF_DST_ CENTRD_ THRSLH_VAL | Default threshold for outlier rule for Distance from Centroid rule. This value can be changed by type criteria. | N | .95 |
| CIS | DEF_PORTIO_ OUTLR_ THRSLH | Default threshold for outlier rule for % cluster size below a threshold rule. This value can be changed by type criteria. | N | 5 |
| CIS | INSIGHT_BI_ SALES_AMT_ MARGIN | Flag to show sales BI based on Sales Amount and Margin | N | N |
| CIS | INSIGHT_BI_ SALES_AMT_ UNIT | Flag to show sales BI based on Sales Amount and Units | N | Y |
| CIS | MAX_HIST_ WEEK_CNT | The maximum number of weeks which should be selectable by the UI when processing historic data | N | 104 |
| CIS | MAX_ITEMS_ IN_GRAPH_ CLUSTER_ DETAIL | Maximum number of clusters to be displayed in Cluster Details graph. | N | 12 |
| CIS | MAX_ITEMS_ IN_GRAPH_ CLUSTER_LIST | Maximum number of clusters to be displayed in Cluster List graph. | N | 20 |
| CIS | MNG_RUN_NO_ WKS | ODisplay run for past n weeks | N | 26 |
| CIS | PERF_ATTR_ TOPN_COUNT | The number of attribute values to be used per product category for performance based clustering | N | 3 |
| CIS | PERF_CIS_ APPROACH | The approach to use for performance-based clustering. Available options are CDT and DT. | N | CDT |
| CIS | PERF_NUM_ WEEKS_FOR_ SLS_SHARE | The number of weeks to be used when calculating the sales share for the product attributes. | N | 16 |
| CIS | PERF_NUM_ WEEKS_FOR_ TOPN_CALC | The number of weeks to be used when identifying the top N attributes. | N | 16 |

*Table 5–6 (Cont.) Advanced Clustering*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| CIS | SELECT_ALL_MERCH_NODES | Flag to identify if all (or only first) merchandise node(s) to be selected by-default | N | N |
| CIS | SUMM_CAL_ALL_LVL | Flag to identify whether performance summarization is allowed at all available calendar levels. | N | N |
| CIS | SUMM_MERCH_ALL_LVL | Flag to identify whether performance summarization is allowed at all available merchandise levels. | N | N |

## Generic Configuration for Store Clustering

Table 5–7 lists the store clustering parameters.

*Table 5–7 Store Clustering Parameters*

| Parameter Name | Parameter Code | Description | Value |
|---|---|---|---|
| MERCH_HIER_TYPE | MERCH_HIER1 | Hierarchy type available for selection in UI - for Primary hierarchy | 1 |
| MERCH_HIER_TYPE | MERCH_HIER2 | Hierarchy type available for selection in UI - for alternate product hierarchy | 5 |
| LOC_HIER_TYPE | LOC_HIER1 | Hierarchy type available for selection in UI - for primary location hierarchy | 2 |
| LOC_HIER_TYPE | LOC_HIER2 | Hierarchy type available for selection in UI - for alternate hlocation hierarchy | 6 |

## Advanced Algorithm Settings

Table 5–8 lists the advanced algorithm settings.

*Table 5–8 Advanced Algorithm Settings*

| Category | Name | Configuration Description | Table |
|---|---|---|---|
| Advanced Algorithm | Flexibility to update algorithm properties | Change default distance metric or scaling methods. For example, update distance metric, scaling methods, and default properties of the algorithm. | Advanced Algorithm Settings: cis_algorithm_attr |
| Advanced Algorithm | Flexibility to override algorithm property by each criteria. | Can have Euclidean distance and seasonality can have correlation. Override default settings of the algorithm by criteria. For example, have the Euclidean distance metric for consumer profile criteria and have correlation metric for criteria having seasonality or time-driven attributes. | Advanced Algorithm: cis_bus_obj_tcrit_algo_attr |

### Outlier and New Store Rules

Outlier rules and new store rules can be configured by defining Oracle types.

*Table 5–9 Outlier and New Store Rules*

| Category | Name | Configuration Setting | Table |
|---|---|---|---|
| Advanced | Flexibility to enable or disable outlier rules to indicates outliers in the cluster set. | Example: Supported rules are "X Percent Less Than Total Size" or distance from centroid greater than x value.<br>Update statement to enable or disable a record.<br>UPDATE cis_outlier_rules<br>SET current_flg = 'N', | cis_outlier_rules |
| Advanced | Flexibility to allocate new stores or stores with poor profiles. | Example: Supported rules are:<br>Post Algorithm:<br>Assign to the largest cluster<br>Assign to the cohesive cluster<br>Assign to like location | cis_newstr_rules |

### ORASE Advanced Clustering SQL Reference

Table 5–10 contains all the possible attributes in the clustering process.

*Table 5–10 CIS_TCRITERIA_ATTR*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | FK for this table. | N |
| NAME | VARCHAR2(50) | Name for the criteria attribute. | N |
| DESCR | VARCHAR2(200) | Additional descriptive text for the criteria attribute. | Y |
| SRC_ENTITY_NAME | VARCHAR2(30) | Name of table or view that holds the attribute information and value coming from RA or an external system. | Y |
| SRC_COLUMN_NAME | VARCHAR2(30) | Name of the column in RA or external system table or view. | Y |
| SRC_ENTITY_ALIAS | VARCHAR2(30) | Alias of the column in the RA or external system table or view. | Y |
| ATTR_TYPE_ID | NUMBER(10) | Type of the attribute. | N |
| CURRENT_FLG | VARCHAR2(1) | A flag to indicate whether this row is the most recent row (Y) or if it a historical row (N). | Y |
| CREATED_BY_USER | VARCHAR2(255) | User who created the record. | N |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether the row is considered deleted (Y) or not (N). | Y |
| CREATED_ON_DT | TIMESTAMP(6) | Record creation date. | N |
| CHANGED_BY_USER | VARCHAR2(255) | User who last changed the record. | Y |
| CHANGED_ON_DT | TIMESTAMP(6) | Last changed date. | Y |

*Table 5–10 (Cont.) CIS_TCRITERIA_ATTR*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BUS_OBJ_ATTR_MD_ID | NUMBER(10) | Business object attribute MD ID. | Y |
| UI_DISPLAY_NAME | VARCHAR2(200) | The value to display in the UI when showing this criteria attribute. | Y |
| DISPLAY_FORMAT_ID | NUMBER(10) | Foreign key to the RSE_DISPLAY_FORMAT table, to enable control of how the attribute should be formated for display. | Y |
| TCRITERIA_ATTR_SRC_XREF_ID | NUMBER(10) | Field contains the reference to the PK of the of the object which this attribute definition belongs to.  For example, if an attribute is representative of a specific consumer segment, then this column will contain the FK to that consumer segment. | Y |

### How to Insert, Update, or Disable a Field

A few routines can perform maintenance of the data in the CIS_TCRITERIA_ATTR and CIS_BUS_OBJ_TCRITERIA_ATT_XREF tables. These routines use the information defined in applicable source tables to set up fields in these tables.A table called CIS_TCRITERIA_SRC_XREF can also help define what attributes should be configured for the different type criteria.

The first task to perform to customiz the different attributes for the different type criteria is to modify the data in CIS_TCRITERIA_SRC_XREF so that it identifies what data sources should be used for a given business object, objective, and type criteria. A default configuration is provided as seed_data for this table. The maintenance routines use the data in this table to determine what attributes should be used by a given business object, objective, and type criteria.

The data in CIS_TCRITERIA_SRC_XREF can be manipulated by reloading data from the SQL*Loader control file, or it can be manipulated directly with SQL. To load a new copy of the control file, you must first update all existing rows so that they are deleted (UPDATE cis_tcriteria_src_xref SET delete_flg = 'Y'; ). Then you must load a modified version of the file via SQL*Loader ( SQLLDR USER/PASSWORD@DB control=cis_tcriteria_src_xref.ctl).

The next step is to run the maintenance routines to populate the attributes into the appropriate tables. These can be performed by executing the following UNIX scripts:

To load consumer segment attributes, use:

- cis_conseg_attr_maint.ksh
- exec cis_maint_util.tcriteria_attr_cs_maint;

To load performance attributes, use:

- cis_perf_attr_maint.ksh

To load product attributes, use:

- cis_prod_attr_maint.ksh

To load store location attributes, use:

- cis_store_attr_maint.ksh

### Table name: cis_bus_obj_tcriteria_att_xref

Description: CIS_BUS_OBJ_TCRITERIA_ATT_XREF

After the above maintenance routines are complete, you can manually adjust the data in CIS_BUS_OBJ_TCRITERIA_ATT_XREF (BOTAX) in order to customize which attributes should be used and how. When adjusting the data in CIS_BUS_OBJ_TCRITERIA_ATT_XREF, consider the following:

- Any attribute that is a participating attribute (PARTICIPATING_ATTR_FLG =Y) with a non-zero weight (DFLT_ATTR_WEIGHT) will be considered a primary attribute.
- An attribute that you do not want to use can be marked as deleted by setting its DELETE_FLG=Y.
- An attribute that is not participating, but is informational (INFORMATIONAL_ATTR_FLG=Y) cannot be used by the clustering algorithm, but is available for review within the UI.
- The ATTR_SEQ column facilitates the ordering of attributes within the UI so that they can be presented in a desired sequence. The PL/SQL routine, cis_maint_util.tcriteria_attr_seq_maint, can set the values according to the selection of participating and primary attributes.
- When maintaining data in this table, you should join this table to the CIS_TCRITERIA_ATTR (TCA) table by TCA.ID = BOTAX.TCRITERIA_ATTR_ID. This helps you identify the attribute that is being manipulated.
- After performing any maintenance of the data in this table, you must make sure that the attribute weights are correctly set. You can run the cis_maint_util.updt_attr_wgts routine to get the weights to total 100%.

To enable a field:

update cis_bus_obj_tcriteria_att_xref, set participating_attr_flg = 'Y', and informational_attr_flg = 'N' where id = :id

To change the sequence of the attribute in the UI:

update cis_bus_obj_tcriteria_att_xref set attr_seq = 10 where id = :id

Sample WHERE clause components (note that a join with cis_tcriteria_attr is necessary to check CURRENT_FLG and DELETE_FLG in that table as well):

from CIS_BUS_OBJ_TCRITERIA_ATT_XREF cbotax, CIS_TCRITERIA_ATTR cta

- where cbotax.BUSSOBJ_ID=1 and cbotax.OBJECTIVE_ID=3 and cbotax.TYPE_CRITERIA_ID in (1) and cbotax.TCRITERIA_ATTR_ID=cta.ID
- and nvl(cbotax.CURRENT_FLG, 'Y') = 'Y' and nvl(cbotax.DELETE_FLG, 'N') = 'N'
- and nvl(cta.CURRENT_FLG, 'Y') = 'Y' and nvl(cta.DELETE_FLG, 'N') = 'N'
- and cta.SRC_COLUMN_NAME='SLS_AMT');

### Table name: cis_tcriteria_src_xref

To delete data, you can update values in the table:

UPDATE cis_tcriteria_src_xref

SET delete_flg ='Y'

- WHERE bussobj_id = :bussobj_id
- AND objective_id = :objective_id
- AND type_criteria_id = :type_criteria_id
- AND src_entity_name = :src_entity_name;

To update data so that a source of attributes is available as a participating or information attribute, you can update the data like this:

UPDATE cis_tcriteria_src_xref

SET participating_attr_flg = :participating_attr_flg,

Informational_attr_flg = :informational_attr_flg

- WHERE bussobj_id = :bussobj_id
- AND objective_id = :objective_id
- AND type_criteria_id = :type_criteria_id
- AND src_entity_name = :src_entity_name;

#### Table name: cis_tcriteria_attr

Description: CIS_TCRITERIA_ATTR

Most maintenance of this data should occur through the cis_maint_util database package, so there should be no need to add rows to this table. However, an attribute can be removed by an update like this:

UPDATE cis_tcriteria_attr SET DELETE_FLG='Y' WHERE id=:id;

To change a label (note that label changes in UI-supplied XLF files are required as well):

UPDATE cis_tcriteria_attr set ui_display_name = 'Sample label' where id = :id

#### Table name: cis_bus_obj_nested_tcriteria

Description: CIS_BUS_OBJ_NESTED_TCRITERIA

To update value:

```
UPDATE cis_bus_obj_nested_tcriteria
        SET current_flg = 'N',
                delete_flg  = 'Y'
 WHERE bussobj_id = :bussobj_id
       AND objective_id = :objective_id
       AND type_criteria_id = :type_criteria_id
AND chl_type_criteria_id = :chl_type_criteria_id;
```

To make a type criteria available for a nested cluster of another type criteria, the data in this table can be modified by updating the CURRENT_FLG, DELETE_FLG and CHL_CRITERIA_ALLWD_FLG to appropriate values (e.g. CURRENT='Y', DELETE_FLG='N', CHL_CRITERIA_ALLWD_FLG='Y').

#### Table name: cis_effective_period

Description: CIS_EFFECTIVE_PERIOD

To insert value: Use following SQL Loader command to insert records using the .ctl file.

```
SQLLDR USER/PASSWORD@DB control=cis_effective_period.ctl
```

To insert using regular insert:

```
INSERT INTO cis_effective_period
  (id, descr, start_period_id, end_period_id, bussobj_id, objective_id, type_
criteria_id, active_flg)
VALUES
```

```
  (1, 'Winter Jan 12', 1005040907, 1005040937, 1, 3, -1, 'Y');
```

#### Table name: cis_bus_obj_tcriteria_xref

Description: CIS_BUS_OBJ_TCRITERIA_XREF

To insert values use following SQL Loader command:

```
SQLLDR USER/PASSWORD@DB control= cis_bus_obj_tcriteria_xref.ctl
```

To insert using regular insert:

```
INSERT INTO cis_bus_obj_tcriteria_xref
  (bussobj_id, objective_id, type_criteria_id, primary_flg, bus_obj_tcriteria_seq,
def_scre_threshold_pct, outlier_rule_id, outlier_attr_value_1, outlier_attr_value_
2, outlier_attr_value_3, newstr_rule_id, disable_attr_wts, current_flg, delete_
flg, created_by_user, created_on_dt, changed_by_user, changed_on_dt)
VALUES
  (1, 3, 1, '', 2, NULL, 1, NULL, NULL, NULL, 2, 'Y', 'Y', '', USER, SYSDATE,
USER, SYSDATE);
```

#### Advanced Algorithm Settings: cis_algorithm_attr

Description: CIS_ALGORITHM_ATTR

To insert values use following SQL Loader command:

```
SQLLDR USER/PASSWORD@DB control= cis_algoritm_attr.ctl
```

To insert using regular insert:

```
INSERT INTO cis_algorithm_attr
  (id, NAME, keyword_name, descr, algorithm_attr_type_id, dflt_value, attr_seq,
cis_algorithm_id, user_provided_flg, current_flg, delete_flg, created_by_user,
created_on_dt, changed_by_user, changed_on_dt)
VALUES
  (1, 'Lambda Stop', 'lambdaStop', 'Given the original lambda, calculate how many
epochs need to run (Should be specific to Cluster Criteria)', 4, '.1', 1, 1, 'N',
'Y', 'N', USER, SYSDATE, USER, SYSDATE);
```

To delete value:

```
DELETE FROM cis_algorithm_attr WHERE id=:id;
```

#### Advanced Algorithm: cis_bus_obj_tcrit_algo_attr

Description: CIS_BUS_OBJ_TCRIT_ALGO_ATTR

*Table 5–11 CIS_BUS_OBJ_TCRIT_ALGO_ATTR*

| Field | Description |
|---|---|
| ALGORITHM_ATTR_ID | FK for the algorithm attribute table. |
| BUSSOBJ_ID | FK for the business object objective. |
| OBJECTIVE_ID | FK for the clustering objective. |
| TYPE_CRITERIA_ID | Type Criteria ID. |
| NAME | This is the name of the clustering algorithm attribute. There are some specific attributes per algorithm. For example, BaNG uses attributes such as distance metric, converge factor, and number of iterations. |

*Table 5–11 (Cont.) CIS_BUS_OBJ_TCRIT_ALGO_ATTR*

| Field | Description |
|---|---|
| KEYWORD_NAME | This field contains the attribute or property name recognized internally by the application code. This value is used within the code to identify the property and must be provided with the correct case and format recognized within the code. |
| DESCR | Additional descriptive text for this algorithm attribute. |
| ALGORITHM_ATTR_ TYPE_ID | Algorithm attribute type ID. |
| DFLT_VALUE | Default value to be used for the attribute. |
| ATTR_SEQ | Sequence in which the attribute should be processed or presented. |
| CIS_ALGORITHM_ID | This is the algorithm ID. This field is populated only for algorithm specific properties or attributes; otherwise, this field is null. |
| USER_PROVIDED_FLG | This flag = 'Y' indicates the user must set this value using the UI while creating the scenario configuration. |
| CURRENT_FLG | A flag to indicate whether this row is the most recent row (Y) or if it a historical row (N). |
| DELETE_FLG | A flag to indicate whether the row is considered deleted (Y) or not (N). |
| CREATED_BY_USER | User who created the record. |
| CREATED_ON_DT | Record creation date. |
| CHANGED_BY_USER | User who last changed the record. |
| CHANGED_ON_DT | Last changed date. |

To update statement to enable or disable a record:

```
UPDATE cis_bus_obj_tcrit_algo_attr
   SET current_flg = 'N',
       delete_flg  = 'Y'
 WHERE algorithm_attr_id = :algorithm_attr_id
   AND bussobj_id = :bussobj_id
   AND objective_id = :objective_id
   AND type_criteria_id = :type_criteria_id;
```

## ASO Configuration

ASO has configurable parameters, listed in Table 5–12, " ASO Configuration Parameters". All have default values, and values chosen as a reasonable starting value, but are configurable and editable by the administrator. In many cases, if the user does not select a value for a particular field, it will default to the value set in this table.

Note that some parameters must be initialized at setup. See Table 5–12. Many of these items that require initialization cannot be configured later. See *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1) for the complete list of which configurations are configurable, updateable by the application, and required at initialization.

The following are the major categories of ASO parameters:

- Alert settings (conditions in which ASO triggers a user alert).

- ASO objective and constraints settings (the settings determining units and default settings for constraints).
- Demand and replenishment settings (the settings for demand and replenishment models).
- Demand transference and service level control settings.
- Gurobi solver parameters (these require expertise in optimization and LP solvers to adjust).
- Default ranges (minimum and maximum values for capacity, height, facings, blocks, and service levels).
- Name, column, and label settings for summary reports.

*Table 5–12 ASO Configuration Parameters*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| SO | ALRT_LESS_THAN_PCT_USED_SPACE | An alert will be triggered if the run optimization results use less space than the value specified by this global parameter. | N | 0.8 |
| SO | ALRT_LESS_THAN_SERVICE_LEVEL_AMT | An alert will be triggered if the run optimization results have a sales service level lower than the value specified by this global parameter. | N | 0.8 |
| SO | ALRT_LESS_THAN_SERVICE_LEVEL_QTY | An alert will be triggered if the run optimization results have a quantity service level lower than the value specified by this global parameter. | N | 0.8 |
| SO | ALRT_MORE_THAN_CNT_PRODUCT_DROPPED | An alert will be triggered if the run optimization results dropped more products than the value specified by this global parameter. | N | 10 |
| SO | ALRT_MORE_THAN_PCT_PRODUCT_DROPPED | An alert will be triggered if the run optimization results dropped a percent of product higher than the value specified by this global parameter. | N | 0.2 |
| SO | ALRT_NO_FEASIBLE_SOLUTION | An alert will be triggered if the run optimization results have no results. | N | 0 |
| SO | ALWAYS_REVIEW_MAPPING_RES_FLG | Default=N. A Y flag indicates a user mapping review is always required (regardless of results or errors). N triggers a review base on other flags and conditions. | N | N |
| SO | CAPACITY_RANGE_UNITS | Capacity range units used by ASO Solver. This parameter value maps to a CRU row with this value ID within so_prod_constr_range_values table. | N | 25 |
| SO | DEFAULT_APPL_USER | User identifier to be used for batch activities that require user tracking. | N | SO_BATCH_USR |
| SO | DEFAULT_BAY_MERGE_CONSTR_FLG | Default indicator for the use of merging bays constraint. | N | N |
| SO | DEFAULT_BLOCKING_CONSTR_FLG | Default indicator for the use of blocking constraint. | N | Y |

*Table 5–12 (Cont.) ASO Configuration Parameters*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| SO | DEFAULT_ OBJECTIVE_ FUNC_ID | Internal ID that identifies the default objective function to use for optimization locations. | N | 1 |
| SO | DEFAULT_ SPACING_ CONSTR_FLG | Default indicator for use of spacing constraint. | N | Y |
| SO | DEFAULT_ USABLE_ SPACE_ CONSTR_FLG | Default flag indicating if space constraint should be used. | N | N |
| SO | DEFAULT_ USABLE_ SPACE_ CONSTR_PCT | Default usable space constraint percentage. | N | 1 |
| SO | DEMAND_ DISTRIBUTION | Demand distribution used by ASO Solver. | N | Normal |
| SO | DFLT_ ANALYTICAL_ THETA | Analytical parameter to calculate tolerance in a blocking diagram is set to theta * (min width of items in a group). | N | 0.1 |
| SO | DFLT_ HORIZONTAL_ BLOCKING_FLG | A Y value for this flag indicates the analytics that combining adjacent attribute blocks should be done (when possible). | Y | Y |
| SO | DFLT_REPL_ CASEPACK | Default replenishment parameter for casepack. | N | 1 |
| SO | DFLT_REPL_ FACINGS_LIFT | Default facing lift. | N | 0 |
| SO | DFLT_REPL_ SHELF_PARAM | Default shelf replenishment parameter. | N | 0 |
| SO | DFLT_REPL_ SHELF_TT | Default replenishment type. | N | 2 |
| SO | DFLT_REPL_ STDEV_ BOOSTER | Default standard deviation booster. | N | 0 |
| SO | DFLT_REPL_ TYPE | Default replenishment type. | N | 2 |
| SO | DFLT_SHELF_ THICKNESS | Default shelf thickness that is used by the POG-shelf interface to create the initial bottom shelf for empty shelf fixtures. | Y | 1 |
| SO | DT_ DAMPENING_ FACTOR | Factor to dampen the DT effects for ASO. | N | 0.5 |
| SO | EMPTY_BLOCK_ CONSTR_ALRT | Alert internal ID for empty visual guideline group alert. | N | 1 |
| SO | END_URL | Application Login End URL. | N | /faces/oracle/retail/rse/so/fe/view/page/SpaceOptimizationHome.jspx |
| SO | GUR_ CLIQUECUTS | Gurobi parameter. Clique cut generation. | N | -1 |

*Table 5–12 (Cont.) ASO Configuration Parameters*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| SO | GUR_COVERCUTS | Gurobi parameter. Cover cut generation. | N | -1 |
| SO | GUR_FLOWPATHCUTS | Gurobi parameter. Flow path cut generation. | N | -1 |
| SO | GUR_HEURISTICS | Gurobi parameter. Turn MIP heuristics up or down. | N | 0.05 |
| SO | GUR_IMPLIEDCUTS | Gurobi parameter. Implied bound cut generation. | N | -1 |
| SO | GUR_MIPFOCUS | Gurobi parameter. Set the focus of the MIP solver. | N | 0 |
| SO | GUR_MIPSEPCUTS | Gurobi parameter. MIP separation cut generation. | N | 0 |
| SO | GUR_MIRCUTS | Gurobi parameter. MIR cut generation. | N | -1 |
| SO | GUR_MODKCUTS | Gurobi parameter. Mod-k cut generation. | N | -1 |
| SO | GUR_NODEMETHOD | Gurobi parameter. Method used to solve MIP node relaxations. | N | 1 |
| SO | GUR_PRESOLVE | Gurobi parameter. Presolve level. | N | -1 |
| SO | GUR_RINS | Gurobi parameter. RINS heuristic. | N | -1 |
| SO | GUR_SYMMETRY | Gurobi parameter. MIP symmetry detection. | N | -1 |
| SO | GUR_TIMELIMIT | Gurobi parameter. Time limit. | N | 90 |
| SO | GUR_THREADS | Gurobi Parameter. Number of allowed threads. | N | 0 |
| SO | GUR_ZEROHALFCUTS | Gurobi parameter. Zero-half cut generation. | N | -1 |
| SO | GV_DAYS_TO_VALIDATE_WO_CHANGES | Number of days without direct changes the validation process will consider data objects for validation. | N | 21 |
| SO | GV_RESULT_DETAIL_LEVEL | Level of detail for each validation that is used to produce the results (DETAIL: rows for every failure or SUMMARY: a row at the data object level). | N | SUMMARY |
| SO | GV_VALIDATION_SECTIONS_TO_RUN | Global validations will be executed for the selected data objects. ASSORTMENT, POG, MAPPING and DS (Display Style). | N | ASSORTMENT_POG_MAPPING_DS |
| SO | INVALID_ATTR_VALUE_CONSTR_ALRT | Alert internal ID for attribute value alert. | N | 3 |
| SO | MAX_CAPACITY_RANGE | Maximum capacity range used by ASO Solver. | N | 80 |
| SO | MAX_HEIGHT_RANGE | Maximum height range used by ASO Solver. | N | 72 |

*Table 5–12 (Cont.) ASO Configuration Parameters*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| SO | MAX_NUMBER_ OF_FACINGS | Maximum number of facings used by ASO Solver. | N | 5 |
| SO | MAX_NUM_ OPT_LOC_ BLOCK | Maximum number of blocks per optimization location. | N | 10 |
| SO | MAX_SHELF_ THICKNESS | Maximum shelf thickness that can be used while doing shelf fixture edits. | Y | 2.5 |
| SO | MIN_ CAPACITY_ RANGE | Minimum capacity range used by ASO Solver. | N | 0 |
| SO | MIN_HEIGHT_ RANGE | Minimum height range used by ASO Solver. | N | 0 |
| SO | MIN_NUMBER_ OF_FACINGS | Minimum number of facings used by ASO Solver. | N | 1 |
| SO | MIN_SHELF_ DEPTH | Minimum shelf depth that can be used while doing shelf fixture edits. The maximum shelf depth is defined by the fixture depth. | Y | 2 |
| SO | MIN_SHELF_ THICKNESS | Minimum shelf thickness that can be used while doing shelf fixture edits. | Y | 0.5 |
| SO | MIN_SHELF_ VERTICAL_GAP | Specific smallest allowable vertical offset (SAVO) value. | Y | 2.5 |
| SO | MNG_ASSORT_ NO_WKS | Display assortments for past *n* weeks. | N | 52 |
| SO | MNG_RUN_NO_ WKS | Display run for past *n* weeks. | N | 52 |
| SO | NUMBER_OF_ SIMULATED_ DAYS | Number of simulated days used by ASO Solver. | N | 1000 |
| SO | OPT_LOC_LVL1_ NAME_STR | This value is used entirely or as a prefix to generate the pogset location and optimization location top level names. | N | All Locations |
| SO | OPT_LOC_LVL2_ NAME_STR | This value is used as a prefix to generate the pogset location and optimization location mid level names. | N | PC_ |
| SO | OPT_LOC_LVL3_ NAME_STR | This value is used as a prefix to generate the pogset location and optimization location bottom level names. | N | SC_ |
| SO | PC_SUM_ CAPRANGE | Capacity Range label for Product Constraints Summary. | N | Set Capacity Range |
| SO | PC_SUM_ ELEVATION | Elevation label for Product Constraints Summary. | N | elevation |
| SO | PC_SUM_ ELEVRANGE | Elevation Range label for Product Constraints Summary. | N | Set Elevation Range |
| SO | PC_SUM_ FACERANGE | Facing Range label for Product Constraints Summary. | N | Set Facing Range |
| SO | PC_SUM_ FACINGS | Facings label for Product Constraints Summary. | N | facings |

*Table 5–12 (Cont.) ASO Configuration Parameters*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| SO | PC_SUM_ INCLUSION | Inclusion label for Product Constraints Summary. | N | Inclusion |
| SO | POGC_SUM_ MERGEBAYS | Merge Adjacent Bays label for POG Constraints. | N | Merge Adjacent Bays |
| SO | POGC_SUM_ PRODSPACE | Use Product Spacing label for POG Constraints. | N | Use Product Spacing |
| SO | POGC_SUM_ USABLESPACE | Set Usable Space label for POG Constraints. | N | Set Usable Space |
| SO | POGC_SUM_VG | Visual Guidelines label for POG Constraints. | N | Use Visual Guidelines |
| SO | POG_SET_LVL1_ NAME_STR | This value is used to generate the name for the top level node on planogram list. | N | All Planograms |
| SO | PRODUCT_ INCLUSION | Product inclusion rule used by ASO Solver. This parameter value maps to a IN row with this value ID within so_prod_constr_ range_values table. | N | 2 |
| SO | PRODUCT_ STACKING_ HEIGHT_LIMIT | Product stacking height limit that is applied as a global setting to all top products (TEMPORARY PARAM). | N | 24 |
| SO | PROD_ATTR_ NAME_ DELIMITER | This value is used as a delimiter between the product name and description and the attribute name and description when setting up POG attributes. A NULL value here will result in no concatenations. | N | - |
| SO | REPLENISHME NT_ FREQUENCY | Replenishment frequency used by ASO Solver. | N | 1 |
| SO | REVIEW_DSF_ ERROR_FLG | A Y flag indicates a user review is required for DSF errors. N lets the process move forward to the next stage using the DSF available. | N | Y |
| SO | REVIEW_ UNMAPPED_ PROD_FLG | A Y flag indicates a user review is required for unmapped products. N lets the process move forward to next stage, eliminating unmapped products. This is not desired for products. | N | Y |
| SO | REVIEW_ UNMAPPED_ STORE_FLG | A Y flag indicates a user review is required for unmapped stores. N lets the process move forward to the next stage, eliminating unmapped stores. | N | Y |
| SO | SO_CAL_HIER_ TYPE | The hierarchy ID to use for the calendar. (Installation configuration) | Y | 10 |
| SO | SO_FISCAL_ CAL_HIER_ TYPE | The hierarchy ID to use for the fiscal calendar. (Installation configuration) | Y | 11 |
| SO | SO_LOC_HIER_ TYPE | The hierarchy ID to use for location. (Installation configuration) | Y | 2 |
| SO | SO_MIN_ SERVICE_LEVEL | Minimum target service level for ASO optimization process. | N | 0.8 |

*Table 5–12 (Cont.) ASO Configuration Parameters*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| SO | SO_PROD_ HIER_LEVEL_ FOR_LEAF_ NODE | Product hierarchy level number for leaf node. | Y | 7 |
| SO | SO_PROD_ HIER_TYPE | The hierarchy ID to use for the product. (Installation configuration) | Y | 1 |
| SO | START_URL | Application Login Start URL. | N | /faces/oracle/retail/rse/so/fe/view/page/SpaceOptimizationHome.jspx |
| SO | STD_ ADJUSTMENT_ COEFFICIENT_1 | Analytical parameter. Demand standard deviation adjustment parameter 1. | N | 0.05 |
| SO | STD_ ADJUSTMENT_ COEFFICIENT_2 | Analytical parameter. Demand standard deviation adjustment parameter 2. | N | 0.19 |
| SO | TOP_SHELF_ STACKING_ HEIGHT_LIMIT | Top shelf stacking height limit that applies as a global setting to all top shelves. | N | 18 |
| SO | TRANSIT_TIME | Default transit time used by ASO Solver. | N | 0 |
| SO | UI_CONFIG_PC_ RENDERED_ COL_7 | UI configuration for product constraints render column 7. Default Y means column is rendered. | N | N |
| SO | UI_CONFIG_PC_ RENDERED_ COL_8 | UI configuration for product constraints render column 8. Default Y means column is rendered. | N | N |
| SO | UI_CONFIG_PC_ RENDERED_ COL_9 | UI configuration for product constraints render column 9. Default Y means column is rendered. | N | N |
| SO | UI_CONFIG_PC_ VISIBLE_COL_1 | UI configuration for product constraints visible column 1. Default Y means column is visible. | N | N |
| SO | UI_CONFIG_PC_ VISIBLE_COL_2 | UI configuration for product constraints visible column 2. Default Y means column is visible. | N | N |
| SO | UI_CONFIG_PC_ VISIBLE_COL_3 | UI configuration for product constraints visible column 3. Default Y means column is visible. | N | N |
| SO | UI_CONFIG_PC_ VISIBLE_COL_4 | UI configuration for product constraints visible column 4. Default Y means column is visible. | N | Y |
| SO | UI_MAX_POG_ CONFIG_ LENGTH | UI configuration for maximum length bound for the Create Lengths pop-up. | N | 600 |
| SO | UI_MIN_POG_ CONFIG_NO_ OF_BAYS | UI configuration for minimum number of bays bound for the Create Lengths pop-up. | N | 1 |
| SO | UI_MAX_POG_ CONFIG_NO_ OF_BAYS | UI configuration for maximum number of bays bound for the Create Lengths pop-up. | Y | N |

*Table 5–12 (Cont.) ASO Configuration Parameters*

| Application Code | Parameter Name | Description | Must Initialize | Example Parameter Value |
|---|---|---|---|---|
| SO | UI_ THRESHOLD_SL | UI configuration for Thresholds Configurable for Service Level formatting. | N | Y |
| SO | UI_ THRESHOLD_ SL_MAX | UI configuration for Thresholds MAX, after which color green is shown. | N | 0.85 |
| SO | UI_ THRESHOLD_ SL_MIN | UI configuration for Thresholds MIN, below which color red is shown. | N | 0.75 |
| SO | UNMAPPED_ ATTR_VALUE_ CONSTR_ALRT | Alert internal ID for unmapped attribute alert. | N | 2 |
| SO | USE_OPT_DT | ASO global indicator for applying DT. | N | N |
| SO | USE_SERVICE_ LEVEL_ CONSTRAINT | ASO global indicator for applying service level constraints. | N | Y |
| SO | WEEK_LENGTH | Length of the week recognized by ASO Solver. | N | 7 |

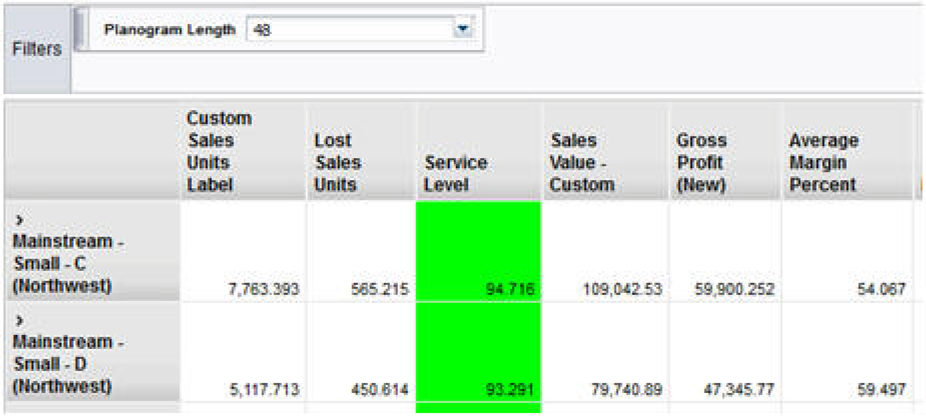
# User Interface Configuration

The CDT, DT, AC, and ASO user interfaces can be configured by adjusting UI labels and date masks. Oracle Metadata Services (MDS) is used for personalization and customization.

## Label Customizations

It is possible to replace most labels and text strings displayed in the user interface with custom labels, if desired. Editing these strings requires extracting and modifying a file within the application archive, which will vary based on the primary language being displayed in the browser. Figure 5–6 shows an example for ASO.

*Figure 5–6 Label Customization*



Planogram Length: 48

| | Custom Sales Units Label | Lost Sales Units | Service Level | Sales Value - Custom | Gross Profit (New) | Average Margin Percent |
|---|---|---|---|---|---|---|
| › Mainstream - Small - C (Northwest) | 7,763.393 | 565.215 | 🟩 94.716 | 109,042.53 | 59,900.252 | 54.067 |
| › Mainstream - Small - D (Northwest) | 5,117.713 | 450.614 | 🟩 93.291 | 79,740.89 | 47,345.77 | 59.497 |

Similar UI label customizations are possible for any ORASE UI application by making corresponding changes to the file and directory names for the application in question.

For ASO, the file name uses the following format:

SOViewControllerBundle.xlf (English only)

SOViewControllerBundle_<lang_code>.xlf (All other languages, where lang_code is the Oracle abbreviated language code, such as SOViewControllerBundle_el.xlf)

Use the following process to make changes to the UI labels in the above files:

1. Locate the directory containing your deployed application archive. For example:

   WLS_DOMAIN_HOME/servers/AdminServer/stage/SO_UI/SpaceOptimization_application1.ear

2. Execute the following command to extract the application interface files:

   ```
   jar -xvf SpaceOptimization_application1.ear soviewcontroller.war
   ```

3. Execute this command to extract the desired label bundle:

   ```
   jar -xvf soviewcontroller.war
   WEB-INF/classes/oracle/retail/rse/so/fe/view/SOViewControllerBundle.xlf
   ```

4. Edit the extracted file by locating the labels that will be changed and entering new text strings. Only the text in the <source></source> tags may be modified; the rest of the file must not be changed or the user interface will not function properly.
5. When your changes have been completed, save the file.
6. Execute the following command to add the file back into the application interface package:

   ```
   jar -uvf soviewcontroller.war
   WEB-INF/classes/oracle/retail/rse/so/fe/view/SOViewControllerBundle.xlf
   ```

7. Execute the following command to add the interface files back into the application archive:

   ```
   jar -uvf SpaceOptimization_application1.ear soviewcontroller.war
   ```

8. Restart your WebLogic server to pick up the modified files.

Metadata for ASO, AC, CDT, and DT can be adjusted as needed by modifying the corresponding database entries for each application. See AC Configuration for examples of MDS customizations.

## Additional Customizations and Extensions

In addition to the configuration points described above, the user interface can be customized by using Oracle JDeveloper and Oracle Metadata Services (MDS). Out-of-the-box, MDS is used by CDT, DT, AC, and ASO standalone (not installed as part of CM under the Fusion Client). It is possible to add matching back end objects as well.

## ETL Customization and Configuration

For ETL customization and configuration, the metadata-driven set of data validation rules and corresponding database types can be extended in the event that additional data validation rules are required above what is already provided.

It is also possible to customize the level at which data validation rules will cause the loader to abort, instead of just bypassing the rows that were invalid. There are two required steps:

1. A new configuration is needed.

2. A new function call is required that references that configuration record, which actually does the data validation.

## Advanced Database Customization

A few options are available for performing metadata adjustments to customize some algorithms used by the applications. It is possible to extend provided database application code so that an alternate implementation is used in order to add some pre processing, some post processing, or to alter an existing algorithm. Since many components of the applications are implemented with database types, these types can be extended, in order to customize the implementation. Although the details of this customization are outside the scope of this document, these tables (RSE_SRVC_CONFIG, RSE_LOAD_SRVC_CFG, DT_EXCLUDE, CDT_EXCLUDE, CIS_NEWSTR_RULES, CIS_OUTLIER_RULES) define details of an implementation to be used for a process. It is possible to alter data in these tables in order to specify a different implementation for a process, without requiring changes to the provided implementations.

## CDT and DT Stage Run Configuration

It is possible to adjust the stages and steps for the CDT and DT data mining activities. It is possible to configure what is being run and to remove and add steps, as desired. The configuration is done through database tables. RSE_PROC_TMPL specifies a computation group, while RSE_PROC_TASK_TMPL specifies the computation stages and steps.

In addition to the supplied sequence of steps, it is possible to define implementation-specific sequences. The actual procedures to be executed are defined in XML files application-context.xml being kept in module's corresponding WAR file. (See WebLogic Configuration.) An example application-context.xml file can be found in Appendix F, "Sample Application Context File (DT)."

## CDT, DT, MBA, and AC Configuration Pluggable Science

The computation and data mining algorithms have several configuration points. All of computational modules have adjustable configuration entries that can affect the various processes and algorithms. Some of these configurations can be adjusted prior to using the applications to adjust performance.

- AC: the clustering algorithm setup is done through XML and metadata.
- DT: the configuration of the modules that execute as a part of the algorithm and the stages and steps are configurable in the database and XML, per above.
- CDT: the configuration of the modules that execute as a part of the algorithm and the stages and steps are configurable in the database and XML, per above.
- MBA: the configuration is in the database.

Data filtering and results pruning for DT and CDT can be adjusted by changing metadata in the appropriate DB tables, per above.

Most database-based batch processes and analytics are implemented in extendable database types, which offer the ability to adjust portions of the algorithms:

Parameter collections enable extensibility through external options.

Parallelism can be adjusted to limit or increase database resource utilization.

Implementation specific pre-processing or post-processing routines can be easily added by extending provided types and adjusting metadata to call alternate implementations.

## 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.

# 6

# Data Integration and Interfaces

This chapter describes data interfaces, ETL scripts, imports and exports, tablespaces, and CM files. It contains the following sections:

## ORASE Data Interfaces

ORASE depends on RADM for much of the data input, and RADM, in turn, acquires data from other sources, such as RMS. Some ASO data are received from CM and other sources through flat file ODI interfaces as defined below. RADM and ORASE are co-deployed on the same Oracle database instance and exchange data in database.

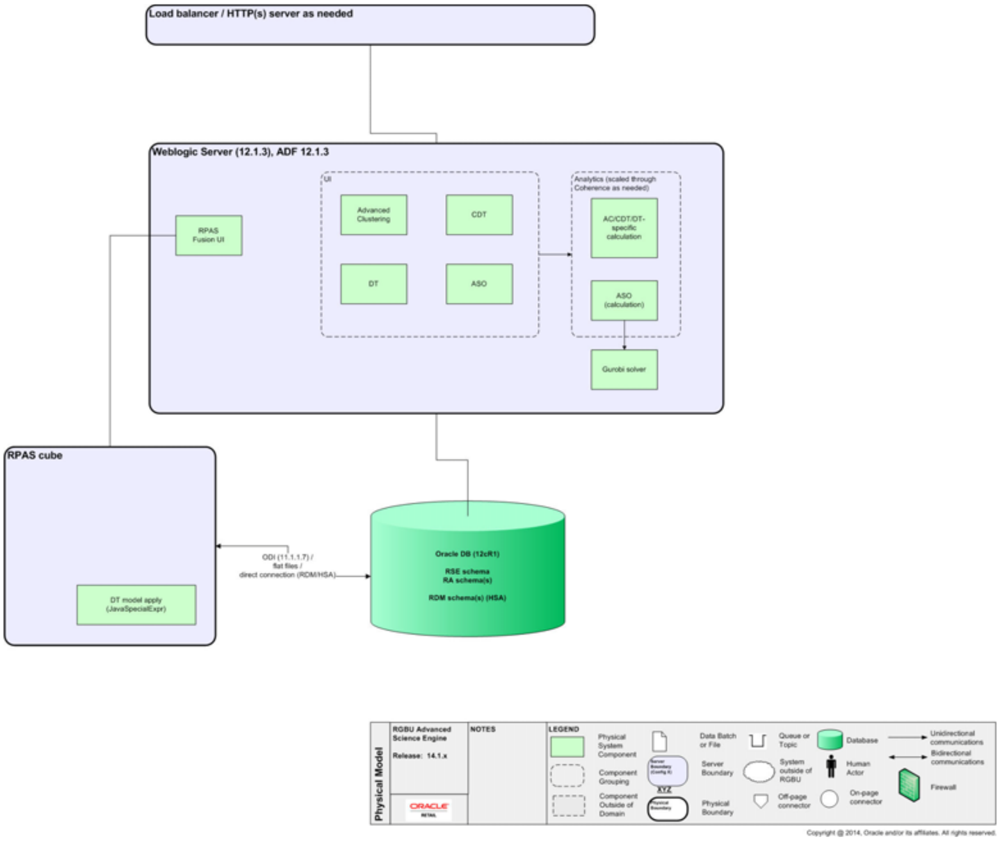
The ORASE interface (see Appendix B, "Database Detail Definitions") defines detailed data requirements for all data to be consumed by ORASE modules and to be sent by ORASE modules to RADM and all subscribing applications.

Subscribing applications include CM, RDF, MSM (or other alternative), and RMS/AIP (or other alternative).

If RMS is installed, the packaged RADM ETL will be used to populate RADM with data from RMS.

Figure 6–1 illustrates the data flow between the related applications. A summary of the main inputs and outputs for ORASE follows.

***Figure 6–1 Data Flow***


Load balancer / HTTP(s) server as needed
Weblogic Server (12.1.3), ADF 12.1.3
RPAS Fusion UI
UI
Advanced Clustering
CDT
DT
ASO
Analytics (scaled through Coherence as needed)
AC/CDT/DT-specific calculation
ASO (calculation)
Gurobi solver
RPAS cube
DT model apply (JavaSpecialExpr)
ODI (11.1.1.7) / flat files / direct connection (RDM/HSA)
Oracle DB (12cR1)
RSE schema
RA schema(s)
RDM schema(s) (HSA)
Physical Model
RGBU Advanced Science Engine
Release: 14.1.x
ORACLE RETAIL
NOTES
LEGEND
Physical System Component
Component Grouping
Component Outside of Domain
Data Batch or File
Server Boundary (Config X)
XYZ
Server Boundary
Physical Boundary
Queue or Topic
System outside of RGBU
Off-page connector
Database
Human Actor
On-page connector
Unidirectional communications
Bidirectional communications
Firewall
Copyright @ 2014, Oracle and/or its affiliates. All rights reserved.

## ORASE Inputs Summary

ORASE and ASO receive input via ETL batch processes:

- From RADM
  - Hierarchies, attributes, sales history (aggregated, daily promotional) and customer-linked transactions history, configuration, price and cost information, segments, item-location ranging, promotion information

ASO uses some of this data from RADM and also receives additional data via file ETL input:

- From CM
  - Assortment and space optimization requests (assortment information), assortment overrides, demand forecasts, placeholder product information, like-items for placeholder products
- From MSM or an alternate source
  - Planograms and product display geometry information
- From RMS, AIP, or an alternate source
  - Product replenishment information

**Note:** Inputs and outputs between ORASE and CM are internal and not detailed in this guide.

## ORASE Outputs Summary

ORASE sends data to CM in the form of ETL files, to RDF in the form of DT API calls, and RA as table views, as follows.

- To CM from DT via files
  - DT parameters, SKU/store, SKU/segment/TA
- To RDF from DT via an API call
  - DT parameters and model apply via an API call
- To CM from CDT via files
  - CDT XML files
- To CM from AC via files
  - Cluster sets
- To RADM from AC via a view
  - Cluster sets

ORASE shares any data residing in the ORASE schema directly with ASO.

ASO makes data available in the form of database views. The consumer application can write custom extracts against these views to retrieve the data.

- To CM via database views and custom extracts
  - Optimized assortments, assortment overrides
- To RMS, AIP, or an alternate source via database views and custom extracts
  - Product replenishment information
- To MSM via custom views
  - Planograms and product display geometry information

Detailed file interface definitions can be found in Appendix B, "Database Detail Definitions".

As part of the implementation, the installation and configuration should already be done as specified. See the *Oracle Retail Modeling Engine Installation Guide*, the *Oracle Retail Assortment and Space Optimization Installation Guide*, and Chapter 5, "Configuration."

## Data Load Batch Scripts Summary

The following classes of batch scripts are provided:

- Data import (including RA to ORASE ETL)
- Data export
- Batch run
- Computation node startup

ORASE and ASO depend on and interact with RA, CM, and RDF. ASO requires additional data from CM, MSM (or other space planning application), and RMS/AIP (or other replenishment application).

To facilitate loading and moving this data, a set of batch scripts is available to orchestrate data import, export, and control batch runs and computation node processing.

These scripts reside on the file system and are created during ORASE installation. The batch scripts rely on the standard Linux technology stack (Linux, Java Virtual Machine, Oracle database client) and environment settings that should be configured as appropriate to allow these scripts to execute properly.

The scripts encompass:

- Data import, which includes running ETL procedures for importing the relevant data from RADM and other systems to ORASE.
- Data export to the subscribing applications.
- Execution of the off-line calculations and a start of the computation scalability nodes, if necessary.

These scripts can be found in the following locations:

- ORASE: <RSE_HOME> /cdm/<cdt, dt, cis>/scripts
- ASO: <RSE_HOME> /so/scripts

## ORASE Data Load and ETL Scripts

ORASE applications require external data from a number of additional sources beyond those that retrieve data from RA. This data resides in flat files as defined in the file interface definition details found in Appendix B, "Database Detail Definitions."

External data is loaded from these flat files as a set of ETL scripts to a set of staging tables and then transformed and loaded to the ORASE or application databases as appropriate. Every load script follows this naming convention:

- <script name>_stg.ctl
- <script name>_stg.ksh
- <script name>_load.ksh

Most ETL is executed as needed by the user. Alternatively, the scripts can be scheduled periodically, based on customer requirements. Some are required to be run periodically.

### Controlling Interface Errors

For all data loaded via a _STG interface table, there is a way to control how errors limits are handled by the load process. The RSE_LOAD_SRVC_CONFIG and RSE_LOAD_VALDT_RULES_CFG tables contain the information related to the load process and their validation rules and are joinable by the RSE_LOAD_SRVC_CONFIG.ID and RSE_LOAD_VALDT_RULES_CFG.LOAD_SRVC_ID columns. The RSE_LOAD_SRVC_CONFIG can be filtered by the NAME column, which is specified in the script that runs the loader.

The validation rules can be set up to fail either if a certain number of failed records occur (via MAX_NUM_ERRORS), or if a percentage of rows in the interface table has been exceeded. If the typical approach to resolving records that fail validation is to remove the row from being processed, then it may be suitable to adjust one or more validation rules so that they have a tolerance for some invalid rows. If rows fail validation but do not exceed the tolerances defined in the validation rules table, then those rows will be deleted from the staging table, the remaining data will be loaded, and the load routine will terminate with a success condition. The rows in the _BAD table remain, so that exceptions can be handled or reported on via a custom post process.

# Retail Analytics Data Load, Control, and ETL

This section provides details about ETL scripts and batch processes.

## RA to ORASE ETL Scripts

Data moves from RADM to ORASE via a set of ETL scripts, as follows:

- Executed manually at initial setup
- Executed manually ad hoc as needed
- Scheduled to run daily or weekly by batch scripts

### At Initial Setup

As described in Chapter 3, "ORASE Installation and Implementation Overview" the RSE common and application-specific configuration and seed data were loaded by executing the *_config.ksh and *_master.ksh scripts. Those steps are mandatory and must be completed before any further data loading.

A summary description of these scripts and control files is provided in this guide. See Oracle *Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1) for details on each script, the interfaces, and batch frequency.

### Loading Additional Sales Transaction Data

The CDT and DT applications require sales transaction data. The seed data load process described in Chapter 3, "ORASE Installation and Implementation Overview" loads the most recent weeks of transaction data by default, as a starting point. This is due to the potential quantity of transaction data. If more transaction data is required, this is the procedure.

The following scripts load sales transaction data from RADM to ORASE:

- rse_sls_trx_setup.ksh
- rse_sls_trx_process.ksh

In addition, the following post-process raw transactions are used:

- rse_wkly_sls_setup.ksh, which aggregates to the weekly level
- rse_wkly_sls_process.ksh
- rse_wkly_sls_seg_setup.ksh, which aggregates to the segment level
- rse_wkly_sls_seg_process.ksh
- rse_fake_cust_setup.ksh, which filters false customer transactions
- rse_fake_cust_process.ksh

All scripts are located in the directory: <RSE_HOME>/common/scripts/bin and are the same scripts that are executed by the cdt_master.ksh during the seed data load process.

These scripts accept a parameter for the number of weeks to be processed. Execute these scripts, choosing an value according the number of weeks desired in ORASE and available in RADM.

For batch loading and post-processing of transactions, these scripts should be scheduled to execute in the same way in a weekly batch with one week to be processed.

### Missing Transaction Data

Sales transaction data is only required by CDT, so if no sales transaction data is available, the other applications can still be used. However, they do require weekly aggregate sales. If RADM does not contain sales transaction data, then weekly aggregate sales data can be gathered using these scripts:

- rse_wkly_sls_stg.ksh for product/location/week aggregates
- rse_wkly_sls_load.ksh scripts
- rse_wkly_sls_seg_stg.ksh for product/location/customer segment/week aggregates
- rse_wkly_sls_seg_load.ksh scripts

The latter of these is required for DT only.

As these aggregations can take time, the suggestion is to stage one week of data at a time into the staging table, and then process that data, and repeat this in a loop for each of the historic weeks to be loaded. This will result in faster processing then if all the weeks are staged at the same time and then processed via a single execution.

The processes are capable of either approach, or any mix of the two approaches. For example, if files are available that contain four weeks per file, then it is acceptable to load the file and process the data in one execution of the load step. You should not attempt this with a large number of weeks in a single file.

Once the weekly aggregates are loaded, some other initial post-processing steps are required. They all should be executed for the same number of weeks to be processed as, and the number should equal the number used when the sales transaction data was loaded (or should equal the number of weeks that were loaded via the aggregate interfaces). The list of initial routines that need to be run are listed in Table 6–1.

*Table 6–1 ORASE Initial Data Setup Routines*

| Application | Setup Script | Process Script | Setup Parameters |
|---|---|---|---|
| CDT | rse_fake_cust_setup.ksh | rse_fake_cust_process.ksh | UPDT_NUM_WEEKS |
| CDT | rse_wkly_sls_ph_aggr_setup.ksh | rse_wkly_sls_process.ksh | UPDT_NUM_WEEKS,<br>FORCE_UPDT_EXISTING |
| AC | rse_wkly_sls_ph_attr_aggr_setup.ksh | rse_wkly_sls_process.ksh | UPDT_NUM_WEEKS,<br>FORCE_UPDT_EXISTING |
| AC | cis_prod_attr_loc_share_setup.ksh | cis_prod_attr_loc_share_process.ksh | UPDT_NUM_WEEKS<br>FORCE_UPDT_EXISTING |
| DT | dt_prod_loc_range_setup.ksh | dt_prod_loc_range_process.ksh | UPDT_NUM_WEEKS<br>FORCE_UPDT_EXISTING |
| DT | dt_loc_range_setup.ksh | dt_loc_range_process.ksh | UPDT_NUM_WEEKS |

After all the historic weeks of data have been processed, it is possible to run the remaining weekly batch routines.

### Recurring Batch Processes Required for All Applications

The processes listed in Table 6–2 must be configured to execute at the frequency listed and apply to all ORASE applications except MBA. They are listed in process order.

Note that many of these processes have a corresponding initial setup script. Only the processing script is listed here. Refer to *Oracle Retail Advanced Science Engine*

*Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1) and find the details on each processing script listed here.

For more details, refer to *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1).

***Table 6–2 Recurring Batch Processes***

| Description | Notes | Frequency | Processing Shell Script |
|---|---|---|---|
| Load the RADM product hierarchy integration ID values. | Product hierarchy interface. This is the first part of the product hierarchy interface. It retrieves the integration IDs and creates new internal IDs as needed for new products. | Daily | rse_prod_src_xref_load.ksh |
| Load the RADM product hierarchy. | Product hierarchy interface. This is the second part of the product hierarchy interface. It retrieves and maintains the product hierarchy descriptions and updates a normalized product hierarchy table. | Daily | rse_prod_hier_load.ksh |
| Create a transitive closure representation of the product hierarchy for the main hierarchy. | Product hierarchy interface. This is a transformation routine that restructures the normalized product hierarchy table data into a transitive closure representation so that it is possible to quickly navigate between any two levels of the product hierarchy. | Daily | rse_prod_tc_load.ksh |
| Transform the product hierarchy to a de-normalized hierarchy representation. | Product hierarchy interface. This is a transformation routine that restructures the normalized product hierarchy table data into a de-normalized representation where there are a fixed number of columns representing each level of the hierarchy. This design is more friendly for reporting displays. | Daily | rse_prod_dh_load.ksh |
| Load the RADM location hierarchy integration ID values. | Location hierarchy interface. This is the first part of the location hierarchy interface. It retrieves the integration IDs and creates new internal IDs as needed for new locations. | Daily | rse_loc_src_xref_load.ksh |
| Load the RADM location hierarchy. | Location hierarchy interface. This is the second part of the location hierarchy interface. It retrieves and maintains the location hierarchy descriptions and updates a normalized location hierarchy table. | Daily | rse_loc_hier_load.ksh |
| Create a transitive closure representation of the location hierarchy. | Location hierarchy interface. This is a transformation routine that restructures the normalized location hierarchy table data into a transitive closure representation so that it is possible to quickly navigate between any two levels of the location hierarchy. | Daily | rse_loc_hier_tc_load.ksh |
| Transform the location hierarchy to a de-normalized hierarchy representation. | Location hierarchy interface. This is a transformation routine that restructures the normalized location hierarchy table data into a de-normalized representation where there are a fixed number of columns representing each level of the hierarchy. This design is more friendly for reporting displays. | Daily | rse_loc_hier_dh_load.ksh |

*Table 6–2 (Cont.) Recurring Batch Processes*

| Description | Notes | Frequency | Processing Shell Script |
|---|---|---|---|
| Load product attribute data from RADM to ORASE's implementation. | This process requires a parameter to control which set of attribute data is to be retrieved. For this process, the parameter should be PRODUCT. | Daily | rse_cda_etl_load.ksh |
| Load the RADM calendar dimension. | The calendar ETL is not a frequently changed dimension, so this routine can be scheduled weekly or less frequently as deemed appropriate. This routine is less likely to be required to run post-application setup. | Weekly | rse_regular_main_ load.ksh |
| Load the RADM fiscal calendar hierarchy. | This calendar ETL is not a frequently changed dimension, so this routine can be scheduled weekly or less frequently as deemed appropriate. It ideally only needs to run as frequently as fiscal calendars are uploaded to RADM. | Weekly | rse_fiscal_main_ load.ksh |

## Recurring Batch Processes Required for ORASE

These processes are either required or optional for ORASE applications (CDT, DT and AC) as listed. They must be configured to execute at the frequency listed and apply to all ORASE applications except MBA. They are listed in Table 6–3 in process order.

Note that many of these processes have a corresponding initial setup script. Only the processing script is listed here. Refer to *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1) and find the details on each processing script listed here.

For more details, refer to *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1).

*Table 6–3 ORASE Recurring Batch Processes*

| Description | Notes | Required by Application | Frequency | Processing Shell Script |
|---|---|---|---|---|
| Load the RADM consumer segment. | Consumer segment hierarchy interface. This is the routine that copies RA's consumer segment data into a suitable table for ORASE to use. This interface is a simple interface without any form of hierarchies. | AC | Daily | rse_conseg_ load.ksh |
| Load the RADM consumer segment location/product allocation data. | This interface copies the RA allocation data that signifies what percentage of available purchasing base exists for each consumer segment for a configured product hierarchy level and store locations. | AC | Daily | rse_conseg_alloc_ load.ksh |
| Load the customer segment hierarchy levels. | This script can be skipped if the data is manually added to the rse_hier_level.ctl file. | CDT, DT | Setup | rse_custseg_level_ load.ksh |
| Load the RADM customer segment integration ID values. | Customer segment interface. This is the first part of the customer segment interface that retrieves data from RA. This process retrieves the integration IDs and the creates new internal IDs as needed for new customer segments. | CDT, DT | Daily | rse_custseg_src_ xref_load.ksh |

*Table 6–3 (Cont.) ORASE Recurring Batch Processes*

| Description | Notes | Required by Application | Frequency | Processing Shell Script |
|---|---|---|---|---|
| Load the RADM customer segment hierarchy. | Customer segment hierarchy interface. This is the second part of the customer segment hierarchy interface. It retrieves and maintains the customer segment descriptions and updates a normalized customer segment hierarchy table. | CDT, DT | Daily | rse_custseg_hier_load.ksh |
| Create a transitive closure representation of the customer segment hierarchy. | Customer segment hierarchy interface. This is a transformation routine that restructures the normalized customer segment hierarchy table data into a transitive closure representation so that it is possible to quickly navigate between any two levels of the customer segment hierarchy. | CDT, DT | Daily | rse_custseg_hier_tc_load.ksh |
| Load the RADM customer segment to customer cross reference data. | Customer segment hierarchy interface. This is the last part of the customer segment hierarchy interface. It retrieves a list of customer IDs from RA that are associated with a customer segment and stores the relationships in a cross reference table so that a customer ID can be classified to its appropriate customer segments. | CDT, DT | Daily | rse_custseg_cust_xref_load.ksh |
| Copy sales transaction data from RADM to ORASE. | If sales transaction data is available in RADM, then this is the starting point of all application's access to sales metrics. All subsequent steps are derived from this data or from aggregations that were derived from this data. | CDT (required)<br>DT (Optional)<br>AC (optional) | Weekly | rse_sls_txn_process.ksh |
| This process performs some ETL of RADM assortment range data into a table specifically designed for DT to use. | This is a required ETL for DT and can later be manipulated via the DT_PROD_LOC_EXCL_STG interface. | DT | Weekly | dt_prod_loc_range_process.ksh |

## Optional Daily Batch Processes for Alternate Hierarchies

Optional daily processes are available if alternate hierarchies are used. If used, they must be configured to execute daily and apply to all ORASE applications except MBA. They are listed in Table 6–4 in process order.

Note that many of these processes have a corresponding initial setup script. Only the processing script is listed here. Refer to *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1) and find the details on each processing script listed here.

For more details, refer to *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1).

*Table 6–4 Batch Processes for Alternate Hierarchies*

| Description | Notes | Processing Shell Script |
|---|---|---|
| Load the RADM CM Group alternate product hierarchy integration ID values. | The CM Group alternate hierarchy allows a customer to have a custom set of product categories be used by all ORASE processes. If the traditional product hierarchy is suitable for grouping related products together, then the routines related to this alternate hierarchy should not be used. Otherwise, this interface will copy the alternate hierarchy integration IDs as defined in RA. | rse_cm_grp_xref_load.ksh |
| Load the RADM CM Group alternate product hierarchy. | Alternate product hierarchy interface. This is the second part of the alternate product hierarchy interface. It retrieves and maintains the alternate product hierarchy descriptions and updates a normalized product hierarchy table. | rse_cm_grp_hier_load.ksh |
| Create a transitive closure representation of the product hierarchy for an alternate hierarchy. | Alternate product hierarchy interface. This is a transformation routine that restructures the normalized product hierarchy table data into a transitive closure representation so that it is possible to quickly navigate between any two levels of the alternate product hierarchy. | rse_prod_tc_load.ksh |
| Transform an alternate product hierarchy to a de-normalized hierarchy representation. | Alternate product hierarchy interface. This is a transformation routine that restructures the normalized alternate product hierarchy table data into a de-normalized representation where there are a fixed number of columns representing each level of the hierarchy. This design is more friendly for reporting displays. | rse_prod_dh_load.ksh |
| Load the RADM trade area hierarchy integration ID values. | The trade area alternate hierarchy allows a customer to have a custom set of location groupings be used for all ORASE analytic processes. If the traditional location hierarchy is suitable for grouping related locations together, then the routines related to this alternate hierarchy should not be used. | rse_trade_area_src_xref_load.ksh |
| Load the RADM trade area hierarchy to the location hierarchy table. | Trade area alternate location hierarchy interface. This is the second part of the trade area alternate location hierarchy interface. It retrieves and maintains the descriptions and updates a normalized location hierarchy table. | rse_trade_area_hier_load.ksh |
| Create a transitive closure representation of the location hierarchy for an alternate hierarchy. | Trade area alternate location hierarchy interface. This is a transformation routine that restructures the normalized location hierarchy table data into a transitive closure representation so that it is possible to quickly navigate between any two levels of the location hierarchy. | rse_loc_hier_tc_load.ksh |
| Transform an alternate location hierarchy to a de-normalized hierarchy representation. | Trade area alternate location hierarchy interface. This is a transformation routine that restructures the normalized location hierarchy table data into a de-normalized representation where there are a fixed number of columns representing each level of the hierarchy. This design is more friendly for reporting displays. | rse_loc_hier_dh_load.ksh |

## ORASE Application Batch Processes

ORASE applications requires a number of additional batch process, listed in Table 6–5, beyond those that retrieve data from RA.

These processes must be configured to execute at the frequency listed and apply to ORASE applications as listed. They are shown in process order.

Note that many of these processes have a corresponding initial setup script. Only the processing script is listed here. Refer to *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1) and find the details on each processing script listed here.

For more details, refer to *Oracle Retail Advanced Science Engine Implementation Guide, Volume 2 - Data Processes and Configuration Variables* (Doc ID 1609804.1).

See Chapter 7, "Market Basket Analysis Overview" for a description of the MBA batch processes.

***Table 6–5 ORASE Batch Processes***

| Description | Notes | Required By Application | Frequency | Processing Shell Script |
|---|---|---|---|---|
| This script executes a process that automates the update of store attribute metadata in the appropriate AC metadata tables. | This script maintains the AC metadata used to define the store attributes in the CIS_TCRITERIA_ATTR and CIS_BUS_OBJ_TCRITERIA_ATT_XREF metadata tables. | AC | As Needed | cis_store_attr_maint.ksh |
| This script executes a process that automates the update of consumer segment attribute metadata in the appropriate AC metadata tables. | This script maintains the AC metadata used to define the consumer segment attributes in the CIS_TCRITERIA_ATTR and CIS_BUS_OBJ_TCRITERIA_ATT_XREF metadata tables. | AC | As Needed | cis_conseg_attr_maint.ksh |
| This script executes a process that automates the update of sales performance attribute metadata in the appropriate AC metadata tables. | This script maintains the AC metadata used to define the sales performance attributes in the CIS_TCRITERIA_ATTR and CIS_BUS_OBJ_TCRITERIA_ATT_XREF metadata tables. | AC | As Needed | cis_perf_attr_maint.ksh |
| This script executes a process that automates the update of product attribute metadata in the appropriate AC metadata tables. | This script maintains the AC metadata used to define the product attribute groups that were loaded via the RSE_PROD_ATTR_GRP_VALUE_STG interface. This routine synchronizes the AC metadata so it contains the relevant data that AC needs. | AC | As Needed | cis_prod_attr_maint.ksh |
| Aggregate sales transaction data to product, location and week. | If sales transaction data has been copied via the CORE_DB_SLS_TXN database service, then this routine will create weekly aggregations of that data. This routine is mutually exclusive with the RSE_SLS_PR_LC_WK load service routine. | DT, AC | Weekly | rse_wkly_sls_process.ksh |
| Aggregate sales transaction data to product, location, customer segment and week. | If sales transaction data has been copied via the CORE_DB_SLS_TXN database service, then this routine will create weekly aggregations of that data. This routine is mutually exclusive with the RSE_SLS_PR_LC_CS_WK load service routine. | DT | Weekly | rse_wkly_sls_seg_process.ksh |

*Table 6–5 (Cont.) ORASE Batch Processes*

| Description | Notes | Required By Application | Frequency | Processing Shell Script |
|---|---|---|---|---|
| Calculate customer ids that are considered fake customers | This is used to search for customer cards such as generic store cards. This routine is only usable if RSE_SLS_TXN data has been copied from RADM. | CDT | Weekly | rse_fake_cust_process.ksh |
| Aggregate weekly sales data to a configured set of product hierarchy levels, location, and week. | This routine is dependent on configurations made in the RSE_AGGR_SRVC_CONFIG_LEVELS table. The cis_perf_attr_maint.ksh script is a prerequisite for this process. | AC | Weekly | rse_wkly_sls_process.ksh |
| Aggregate weekly sales data to a configured set of product hierarchy levels, attribute values, location, and week. | This routine is dependent on configurations made in the RSE_AGGR_SRVC_CONFIG_LEVELS table. The cis_perf_attr_maint.ksh script is a prerequisite for this process. | AC | Weekly | rse_wkly_sls_process.ksh |
| This script calculates product attribute location share metrics for performance-based clustering metrics. | The cis_perf_attr_maint.ksh script is a prerequisite for this process. | AC | Weekly | cis_prod_attr_loc_share_process.ksh |
| This script performs some aggregation of product location assortment range data. | This routine should run after the loading and manipulation of the ranging data from the prior prerequisite steps. | DT | Weekly | dt_loc_range_process.ksh |
| This process executes the baseline sales calculation batch process. | The main input to this process is the RSE_SLS_PR_LC_CS_WK_ and RSE_SLS_PR_LC_WK tables. | DT | Weekly | dt_baseline_process.ksh |
| This script maintains a table of calendar intervals that Demand Transference performs its calculations within. | This script only needs to be executed when new fiscal calendars are loaded, although there is no harm in running routinely. The configuration that drives this process cannot be changed once the process has been executed, as the intervals need to remain uniform. The process does not completely replace all previously defined intervals; therefore, it is important to set the configurations correctly before running this step. If the configurations do need to change, then the tables that relate to this data will need to be reset. | DT | Weekly | dt_updt_mdl_interval.ksh |

*Table 6–5 (Cont.) ORASE Batch Processes*

| Description | Notes | Required By Application | Frequency | Processing Shell Script |
|---|---|---|---|---|
| Demand Transference Model Update. | This script looks for Demand Transference Models that are missing data for new intervals of time and executes the appropriate processes to calculate those intervals and update the Demand Transference Models. This script is recommended to be run weekly, although it can be controlled so that it only processes a portion of the models at each execution. This allows a more even spread of processing resources from one week to another. The use of this script is optional. If the implementation does not desire automatically updated models, then this routine should not be used. However, it is recommended that the models be continuously updated as new data arrives, and this routine accomplishes that. The models retain previously defined decisions, and only adjust the models via relatively small adjustments. | DT | Weekly | dt_updt_model_process.ksh |
| Auto Update escalation paths for CDT. | This routine is needed to ensure that new customer segments or locations are assigned CDTs according to the previously defined escalation rules. Once a location and customer segment has a CDT assigned, it will not change via this process. | CDT | Weekly | cdt_updt_esc_results.ksh |
| Auto Update escalation paths for DT Models. | This script performs routine maintenance on escalation results, so that as new locations and customer segments are added to the system, they can have Demand Transference results applied to them. | DT | Weekly | dt_updt_esc_results.ksh |
| Load product attribute metadata for ASO. This process copies some required values from ORASE tables to be used by ASO's visual guideline feature. | Insert PAG and PAGV to ASO table. | ASO | As needed | so_update_prod_attr_proc.ksh |
| This script prepares a table of clusters to be exported to external applications. | Upon the completion of this, routines that use data from CIS_CLUSTER_SET_EXP_VW can be executed. This is also a prerequisite for RA to obtain the cluster outputs, which RA retrieves via batch processing. | AC | Weekly | cis_prepare_cluster_exp.ksh |

*Table 6–5 (Cont.) ORASE Batch Processes*

| Description | Notes | Required By Application | Frequency | Processing Shell Script |
|---|---|---|---|---|
| This process performs a series of calculations needed to support the DT_RDF or DT_AIP export processes. | This process is a prerequisite for either the DT_RDF or DT_AIP job processor task. | DT | Weekly | dt_export_ prep.ksh |
| This process performs a series of calculations to prepare data for export to the RDF application. | This script does not create any export files, but does prepare data in tables so that subsequent export scripts can export appropriate data. | DT | Weekly | dt_rdf_ export.ksh |
| This process performs a series of calculations to prepare data for export to the AIP application. | This script creates a file to be sent to the AIP application, and outputs all data as processed by the DT_AIP job processor task. | DT | As Needed | dt_aip_ export.ksh |

# ASO Data Import

ASO requires some of the same RA content that ORASE retrieves. It also requires additional inputs from CM and from other sources such as MSM and AIP, as defined in the ORASE Inputs Summary.

For these additional inputs, ASO uses ETL data load scripts that are executed either as needed or in a scheduled batch. These ETL files must adhere to the file standards defined below.

Note that product IDs and merchandise hierarchies in these ETL files must align with those received from RADM.

Interfaces between CM and ORASE are internal and not documented here. To review data received or passed to from CM, reference the Database Staging and Export view respectively. See Appendix B, "Database Detail Definitions" for details.

## MSM (or Similar) to Assortment and Space Optimization Interfaces

Historical planogram information must be loaded into the ORASE database. This information is used to define POG shelf/pegboard dimensions, categories, seasonal information, and product display geometry.

The input file interface follows the Oracle MSM file format so that MSM customers can export historical POG information to files that ASO can import. Customers who use another planogram tool must create the import files to match this interface definition.

It is recommended at implementation to import historical POGs in bulk, rather than on an ad hoc basis. Additional POGs can then be added incrementally as necessary.

**Note:** All units that are used in this interface are predefined at implementation time.

### POG Definition File

This file defines the major characteristics of a POG, including name, category, status, seasonality, and dimensions.

*Table 6–6 POG Definition File*

| Example | Field | Type | Description |
|---|---|---|---|
| POG_WINTER_104-2 | POG_Key | VARCHAR2(80) | External planogram ID. |
| GROCERY_ Beverages-Coffee_02 | POG_Name | VARCHAR2(80) | Planogram name. |
| Fall GROCERY Beverages Coffee 02 | POG_Desc | VARCHAR2(80) | Planogram description. |
| FY2014 | Season_Code | VARCHAR2(30) | Code that identifies the season for which the planogram should be used. |
| Winter | seasonal_attribute | VARCHAR2(30) | Attribute that describes the season of the year for which the planogram should be used. For example, spring, holiday, year-round. |
| 2013-12-21 | effective_start_dt | Date | Earliest date in the year for which the planogram is effective. The format is YYYY-MM-DD. |
| 2014-03-21 | effective_end_dt | Date | Planogram's end date. Last day of the year that the planogram is effective. The format is YYYY-MM-DD. |
| Approved | Status | VARCHAR2(30) | Current planogram status. Approved, Rejected, Pending, or Received. It comes in the feed as Pending.<br>ASO provides a list of valid values for POG status within the configuration data files (so_pog_status.ctl).<br>This description can be customized or translated but the meaning and ID of each status must remain the same since the application uses the IDs for specific purposes.<br>The value provided here must match one of the description values within that file or the row will be rejected. |
| Xmas~2014 | Category_Key | VARCHAR2(80) | POG category key. The second lowest level of POG hierarchy. This value is mandatory. |
| Holiday Items 2014 | Category_Name | VARCHAR2(80) | POG category name. |
| Celebration~220117 | Sub_Category_Key | VARCHAR2(80) | POG sub-category key. The lowest level of POG hierarchy. This value is mandatory. |
| Celebration | Sub_Category_ Name | VARCHAR2(80) | POG sub-category name. |
| Decoration~22 | Dept_Key | VARCHAR2(80) | POG department key. This value is mandatory. |
| Decoration | Dept_Name | VARCHAR2(80) | POG department name. |

*Table 6–6 (Cont.) POG Definition File*

| Example | Field | Type | Description |
|---|---|---|---|
| 144 | Length | NUMBER(18,4) | The total length of a planogram.<br>All units of measure must be provided using the same units, across the whole application.<br>All dimensions for all POG components, product display styles, fixture size, and product positions must use a common unit of measure (for example, inches or centimeters). |
| 36 | Depth | NUMBER(18,4) | The maximum depth of a planogram.<br>All units of measure must be provided using the same units, across the whole application.<br>All dimensions for all POG components, product display styles, fixture size, and product positions must use a common unit of measure (for example, inches or centimeters). |
| 96 | Height | NUMBER(18,4) | The maximum height of a planogram.<br>All units of measure must be provided using the same units, across the whole application.<br>All dimensions for all POG components, product display styles, fixture size, and product positions must use a common unit of measure (for example, inches or centimeters). |

### POG Store File

This file maps a POG to a particular store key. This can be a cluster of actual stores.

*Table 6–7 POG Store File*

| Example | Field | Type | Description |
|---|---|---|---|
| POG_WINTER_104-2 | POG_Key | VARCHAR2(80) | External planogram ID. |
| 56 | Store_Key | VARCHAR2(80) | External store ID. |
| 2013-12-21 | Effective_Start | Date | Start day of the year for which the historical planogram is effective for the store. Format YYYY-MM-DD. |
| 2014-03-21 | Effective_End | Date | End day of the year for which the historical planogram is effective for the store. format YYYY-MM-DD. |

### POG Display Style File

This file lists the display styles used in certain planograms. It lists which display styles are used in a finished POG.

*Table 6–8 POG Display Style File*

| Example | Field | Type | Description |
|---|---|---|---|
| POG_WINTER_104-2 | POG_Key | VARCHAR2(80) | External planogram ID. |
| SCI_DS_1234816 | Display_Style_Key | VARCHAR2(80) | External display style ID. |

## Display Style Orientation File

This file is a cross reference between display style and orientation. This lists the valid orientations for each display style. Each display style must be mapped against at least one orientation.

*Table 6–9 Display Style Orientation File*

| Example | Field | Type | Description |
|---|---|---|---|
| SCI_DS_1236808 | DisplayStyle_Key | VARCHAR2(80) | The external display style ID, matching the value from the POG display style file. |
| Front 0 | Orientation_Key | VARCHAR2(80) | External orientation ID.<br>ASO provides a list of valid orientations within the configuration data files (so_orientation.ctl). The value for orientation_ext_key within that file can be modified or translated to assign different descriptions for each value provided. The description can be changed but the meaning must remain the same since the IDs are already widely used across the application to adjust product dimensions based on the selected orientation.<br>The value provided here must match one of the orientation_ext_key values within that file or the row will be rejected. |
| Y | Default_Flag | VARCHAR2(1) | Y indicates the orientation should be consider as the default for the display style. N indicates the orientation is valid for the display style but is not a default. |

## POG Bay Configuration File

This file provides a list of bays used by the planograms.

*Table 6–10 POG Bay Configuration File*

| Example | Field | Type | Description |
|---|---|---|---|
| POG_FALL_02_BAY_02 | Bay_Key | VARCHAR2(80) | The bay key is associated with certain POG only. The bay's external ID. |
| POG_FALL_02 | POG_Key | VARCHAR2(80) | The planogram's external ID. It must match the POG file ID. |
| 3 | Bay_Sequence | NUMBER(3) | The position of the bay (left to right) within the planogram. |

## Fixture Definition File

This file provides a list of the fixtures that define the planogram.

*Table 6–11 Fixture Definition File*

| Example | Field | Type | Description |
|---|---|---|---|
| SCI_POG_04_2_BAY_01_FXT_01 | Fixture_Key | VARCHAR2(80) | The external fixture ID. |
| Shelf | Fixture_Type | VARCHAR2(80) | Fixture type can be Shelf, Pegboard, or Freezer Chest.<br>ASO provides a list of supported fixture types within the configuration data files (so_fixture_type.ctl). The value for fixture type description on that file can be modified or translated to assign different descriptions for each value provided. The description can be changed but the meaning must remain the same since the IDs are already widely used across the application.<br>The value provided here must match exactly one of the descriptions within that file or the row will be rejected. |
| 24 | Depth | NUMBER(18,4) | The fixture's maximum depth. |
| 74 | Height | NUMBER(18,4) | The fixture's maximum height. |
| 48 | Width | NUMBER(18,4) | The fixture's maximum width. |
| 0.5 | Vertical_Spacing | NUMBER(18,4) | This field is used for pegboard fixture. |
| 0.2 | Horizontal_Spacing | NUMBER(18,4) | This field is used for pegboard fixture. |
| 48 | Max_Length | NUMBER(18,4) | This field is used for pegboard fixture. |
| 42.5 | Capacity_X | NUMBER(18,4) | Freezer length. This field is used for freezer fixture. |
| 19.5 | Capacity_y | NUMBER(18,4) | Freezer depth. This field is used for freezer fixture. |
| 68.5 | Capacity_Z | NUMBER(18,4) | Freezer height. This field is used for freezer fixture. |

### Fixture Configuration File

This file describes the fixture layout in a bay. A fixture can be Shelf, Pegboard, or Freezer Chest.

*Table 6–12 Fixture Configuration File*

| Example | Field | Type | Description |
|---|---|---|---|
| POG_FALL_02_BAY_02 | Bay_Key | VARCHAR2(80) | The external bay ID. It must match the value in the POG Bay file. |
| SCI_POG_04_2_BAY_01_FXT_01 | Fixture_Key | VARCHAR2(80) | The external fixture ID. It must match the ID from the fixture file. |
| 0 | Position_x | NUMBER(18,4) | Fixture position on the X axis relative to the bay. Origin point: bottom, left, back. |
| 0 | Position_y | NUMBER(18,4) | Fixture position on the Y axis relative to the bay. Origin point: bottom, left, back. |
| 0 | Position_z | NUMBER(18,4) | Fixture position on the Z axis relative to the bay. Origin point: bottom, left, back. |

### Display Style Compatibility File

Cross reference file between fixture types and display styles. This lists the fixtures for which the display style is valid.

*Table 6–13 Display Style Compatibility File*

| Example | Field | Type | Description |
|---|---|---|---|
| SCI_DS_1236808 | DisplayStyle_Key | VARCHAR2(80) | The external display style ID, matching the value from the POG display style file. |
| Freezer Chest | Fixture_Type | VARCHAR2(80) | Fixture type can be Shelf, Pegboard, or Freezer Chest.<br>ASO provides a list of supported fixture types within the configuration data files (so_fixture_type.ctl). The value for fixture type description on that file can be modified or translated to assign different descriptions for each value provided. The description can be changed but the meaning must remain the same since the IDs are already widely used across the application.<br>The value provided here must match exactly one of the descriptions within that file or the row will be rejected. |

### Shelf Definition File

This file is provided and required for planograms that include shelf fixtures. It provides the details for each individual shelf in the fixture.

*Table 6–14 Shelf Definition File*

| Example | Field | Type | Description |
|---|---|---|---|
| POG_WINTER_01_BAY_03_FXT_01_SF_3 | Shelf_Key | VARCHAR2(80) | The external shelf ID. |
| 24 | Depth | NUMBER(18,4) | The shelf’s physical depth. |
| 0.8 | Height | NUMBER(18,4) | The shelf’s physical height. This is the thickness of the shelf, not to be mistaken with the space for the product on top of the shelf. |
| 48 | Width | NUMBER(18,4) | The shelf’s width. |

### Shelf Configuration File

This file describes the shelf layout in a fixture (used for shelf fixture only).

*Table 6–15 Shelf Configuration File*

| Example | Field | Type | Description |
|---|---|---|---|
| SCI_POG_04_2_BAY_01 | Bay_Key | VARCHAR2(80) | The external bay ID. |
| SCI_POG_04_2_BAY_01_FXT_01 | Fixture_Key | VARCHAR2(80) | The fixture external ID. |
| SCI_POG_04_2_BAY_01_FXT_01_SF_07 | Shelf Key | VARCHAR2(80) | The shelf external ID. |

*Table 6–15 (Cont.) Shelf Configuration File*

| Example | Field | Type | Description |
|---|---|---|---|
| 0 | Pos_x | NUMBER(18,4) | The origin point on the X axis. |
| 0 | Pos_y | NUMBER(18,4) | The origin point of the Y axis. |
| 61 | Pos_z | NUMBER(18,4) | The origin point on the Z axis. |

### Product Display Style File

Product to display style mapping. It provides a list of display styles available for a specific product.

*Table 6–16 Product Display Style File*

| Example | Field | Type | Description |
|---|---|---|---|
| 1239856 | Product_Key | VARCHAR2(80) | (RSE Core) It must match the merchandise key definition in RSE Core. |
| SCI_DS_1234816 | Display_Style_Key | VARCHAR2(80) | External display style ID, matching the ID from the POG display style file. It links an historical planogram with a specific product. |
| Y | Default Flg | VARCHAR2(1) | Y - Indicates the default display style for a given product.<br>N - Indicates the combination should not be considered as a default.<br>Each product should have one default display style. |

### Display Style Definition File

This file provides the display style product settings and dimensions.

*Table 6–17 Display Style Definition File*

| Example | Field | Type | Description |
|---|---|---|---|
| SCI_DS_1234816 | Display_Style_Key | VARCHAR2(80) | External display style ID, matching the ID from the POG display style file. |
| 1236214 - Folgers 100% Columbian Non-Flavored De-C | Display_Style_Name | VARCHAR2(80) | The name associated with the display style. |
| 1236214 - Folgers 100% Columbian Non-Flavored De-C | Display_Style_Desc | VARCHAR2(80) | The display style description. |
| 3.73 | Depth | NUMBER(18,4) | The dimension is relevant to "Front", "0" orientation. |
| 5.25 | Height | NUMBER(18,4) | The dimension is relevant to "Front", "0" orientation. |
| 7.2 | Width | NUMBER(18,4) | The dimension is relevant to "Front", "0" orientation. |
| 0.2 | Finger_Space_Above | NUMBER(18,4) | The gap between same product above. |
| 0.2 | Finger_Space_Beside | NUMBER(18,4) | The gap between the same product side by side. |
| 0.1 | Finger_Space_ Behind | NUMBER(18,4) | The gap between the same product one in front of the other. |

*Table 6–17 (Cont.) Display Style Definition File*

| Example | Field | Type | Description |
|---|---|---|---|
| 0.5 | Inter_Product_Gap | NUMBER(18,4) | The gap between products. This field captures the gap between different products. |
| 2 | Max_stack | NUMBER(10) | The number of items that can be stacked together. This is equal to 1 if not stackable. |
| 0 | Nesting_Height | NUMBER(18,4) | The product nesting height. The product does not allow nesting if all nesting dimensions are 0. |
| 0 | Nesting_Width | NUMBER(18,4) | The product nesting width. The product does not allow nesting if all nesting dimensions are 0. |
| 0 | Nesting_Depth | NUMBER(18,4) | The product nesting depth. The product does not allow nesting if all nesting dimensions are 0. |
| Maroon | Color | VARCHAR2(30) | The product color, which can be null. |
| 1 | Display_Units | NUMBER(3) | For unit display style it is 1; otherwise, it is greater than 1. Values of null or 0 are converted to 1. |
| Single or Unit | Display_Style_Type | VARCHAR2(80) | A valid display style type, such as Case, Pallet, Single or Unit, and Tray. |

### Shelf Product Configuration File

Describes the product layout of the shelf fixture. Products are always put at the lowest level of equipment; the anchor point locates at the lower left part.

*Table 6–18 Shelf Product Configuration File*

| Example | Field | Type | Description |
|---|---|---|---|
| SCI_DS_1234747 | DisplayStyle_Key | VARCHAR2(80) | The display style external ID. It must match the value in the POG display style file. |
| POG_FALL_02_BAY_02 | Bay_Key | VARCHAR2(80) | The bay external ID. It must match the value in the POG bay file. |
| SCI_POG_04_2_BAY_01_FXT_01 | Fixture_Key | VARCHAR2(80) | The fixture external ID. It must match the value in the POG fixture file. |
| SCI_POG_04_2_BAY_01_FXT_01_SF_06 | Shelf_Key | VARCHAR2(80) | The shelf external ID. It must match the value in the POG shelf file. |
| Front 0 | Orientation_Key | VARCHAR2(80) | The current orientation of this product within the fixture.<br>ASO provides a list of valid orientations within the configuration data files (so_orientation.ctl). The value for orientation_ext_key within that file can be modified or translated to assign different descriptions for each value provided. The description can be changed but the meaning must remain the same since the IDs are already widely used across the application to adjust product dimensions based on the selected orientation.<br>The value provided here must match one of the orientation_ext_key values within that file or the row will be rejected. |

*Table 6–18 (Cont.) Shelf Product Configuration File*

| Example | Field | Type | Description |
|---|---|---|---|
| 24.5 | Pos_x | NUMBER(18,4) | The position of the product within the shelf X axis. |
| 0 | Pos_y | NUMBER(18,4) | The position of the product within the shelf Y axis. |
| 0 | Pos_z | NUMBER(18,4) | The position of the product within the shelf Z axis. |
| 8 | Facing_Quantity | NUMBER(5) | The number of the product's facings displayed on the shelf. |

### Pegboard/Freezer Product Configuration File

Describes the product layout on a freezer/pegboard fixture. Products sare always positioned at the lowest level of equipment; the anchor point at lower left part.

*Table 6–19 Pegboard Product Configuration File*

| Example | Field | Type | Description |
|---|---|---|---|
| SCI_DS_1234711 | DisplayStyle_Key | VARCHAR2(80) | The display style external ID. It must match the value in the POG display style file. |
| SCI_POG_04_2_BAY_01 | Bay_Key | VARCHAR2(80) | The bay external ID. It must match the value in the POG bay file. |
| SCI_POG_04_2_BAY_01_ FXT_01 | Fixture_Key | VARCHAR2(80) | The fixture external ID. It must match a value within the POG fixture file. |
| Back 90 | Orientation_Key | VARCHAR2(80) | Current orientation of this product.<br>ASO provides a list of valid orientations within the configuration data files (so_ orientation.ctl). The value for orientation_ ext_key within that file can be modified or translated to assign different descriptions for each value provided. The description can be changed but the meaning must remain the same since the IDs are already widely used across the application to adjust product dimensions based on the selected orientation.<br>The value provided here must match one of the orientation_ext_key values within that file or the row will be rejected. |
| 24.5 | Pos_x | NUMBER(18,4) | The origin position of the product within the pegboard/freezer: bottom, left, back. |
| 0 | Pos_y | NUMBER(18,4) | The position of the product within the pegboard/freezer Y axis. |
| 0 | Pos_z | NUMBER(18,4) | The position of the product within the pegboard/freezer Z axis. |
| 8 | Facing_Quantity | NUMBER(5) | The number of the product's facings displayed in the pegboard/freezer. |

### Store Custom Defined Attributes

This file provides for every loaded store/historical POG the ability to add a combination of generic user-defined POG attributes: 10 numeric; 5 dates; 5 text, 5 percent %. These attributes are presented within the review results UI. Aggregation and handling of these attributes is adjusted based on the data type and level of aggregation for which the data is presented within the UI.

*Table 6–20 Pegboard Product Configuration File*

| Example | Field | Type | Description |
|---|---|---|---|
| POG_WINTER_104-2 | Pog_Key | VARCHAR2 (80 CHAR) | Planogram unique identifier |
| 56 | Store_Key | VARCHAR2 (80 CHAR) | This is the external store ID, known and shared across applications. |
| 22.4353 | Attr_Num_Value_1 | NUMBER (18,4) | Generic numeric attribute |
| 1.5004 | Attr_Num_Value_2 | NUMBER (18,4) | Generic numeric attribute |
| 18.211 | Attr_Num_Value_3 | NUMBER (18,4) | Generic numeric attribute |
| 7.4314 | Attr_Num_Value_4 | NUMBER (18,4) | Generic numeric attribute |
|  | Attr_Num_Value_5 | NUMBER (18,4) | Generic numeric attribute |
| 47.7185 | Attr_Num_Value_6 | NUMBER (18,4) | Generic numeric attribute |
| 19.8553 | Attr_Num_Value_7 | NUMBER (18,4) | Generic numeric attribute |
| 26.1975 | Attr_Num_Value_8 | NUMBER (18,4) | Generic numeric attribute |
|  | Attr_Num_Value_9 | NUMBER (18,4) | Generic numeric attribute |
|  | Attr_Num_Value_10 | NUMBER (18,4) | Generic numeric attribute |
| 11-DEC-2013 | Attr_Date_Value_1 | DATE | Generic date attribute |
|  | Attr_Date_Value_2 | DATE | Generic date attribute |
| 5-DEC-2013 | Attr_Date_Value_3 | DATE | Generic date attribute |
| 18-DEC-2013 | Attr_Date_Value_4 | DATE | Generic date attribute |
|  | Attr_Date_Value_5 | DATE | Generic date attribute |
| Special Store Opening | Attr_String_Value_1 | VARCHAR2 (80 CHAR) | Generic text attribute |
|  | Attr_String_Value_2 | VARCHAR2 (80 CHAR) | Generic text attribute |
|  | Attr_String_Value_3 | VARCHAR2 (80 CHAR) | Generic text attribute |
|  | Attr_String_Value_4 | VARCHAR2 (80 CHAR) | Generic text attribute |
|  | Attr_String_Value_5 | VARCHAR2 (80 CHAR) | Generic text attribute |
| 0.7386 | Attr_Pct_Value_1 | NUMBER (5,4) | Generic percentage attribute |
| 0.3725 | Attr_Pct_Value_2 | NUMBER (5,4) | Generic percentage attribute |
| 0.9714 | Attr_Pct_Value_3 | NUMBER (5,4) | Generic percentage attribute |
|  | Attr_Pct_Value_4 | NUMBER (5,4) | Generic percentage attribute |
| 0.7109 | Attr_Pct_Value_5 | NUMBER (5,4) | Generic percentage attribute |

## AIP/RO (or Other Replenishment) to Assortment and Space Optimization Interfaces

### Assortment and Space Optimization Replenishment Parameters File

This file provides the replenishment parameters at the product/location level. Each product/location must have a corresponding replenishment record.

*Table 6–21 Assortment and Space Optimization Replenishment Parameters File*

| Example | Field | Type | Description |
|---|---|---|---|
| 1234582 | Product_Key | VARCHAR2(80) | This is the external ID that is known and shared across applications. For placeholder products, this field contain the CM placeholder product key. |
| 56 | Location_Key | VARCHAR2(80) | This is the external store ID, known and shared across applications. |
| 6 | casepack | NUMBER(18,4) | Product casepack for a given store. |
| 2 | replenishment_freq | NUMBER(18,4) | Replenishment frequency (RF) = number of replenishments to the shelf per week. |
| 2 | replenishment_type | NUMBER(10) | Replenishment source/type - two options: 1 = from DC/vendor, 2 = from back room. |
| 0 | transit_time | NUMBER(10,2) | Transit time (TT) = number of days it takes an orde.r to go from source (DC or backroom) to shelf. |
| 2 | shelf_ replenishment_tt | NUMBER(10) | Shelf replenishment trigger type - 3 options: 1 = cover demand over replenishment period + transit time<br>2 = replenishment when inventory gets to a target percentage of capacity<br>3 = replenishment when a casepack can fit |
| 95 | shelf_ replenishment_ param | NUMBER(18,4) | Shelf replenishment parameter (currently only applies for option 2). |
| 0.05 | stdev_booster | NUMBER(10,6) | Standard deviation booster (number greater than or equal to 0, makes sense to limit to 1). |
| 7 | days_of_sales_per_ wk | NUMBER(3,2) | Days of sales per week (number between 1 and 7). |
| 0 | facings_lift | NUMBER(5,4) | Facing lift parameter. |

### Assortment and Space Optimization POG to Assortment Mapping File

This file contains the POG hierarchy to assortment product mapping information. This data is used to identify which POG should be used for each product.

*Table 6–22 Assortment and Space Optimization POG to Assortment Mapping File*

| Example | Field | Type | Description |
|---|---|---|---|
| Decoration~22 | pog_dept_key | VARCHAR2(80) | This is the POG dept key. This is a POG hierarchy external key known to the external source. This is a mandatory value. |
| Xmas~2201 | pog_category_key | VARCHAR2(80) | This is the POG category key. This is a POG hierarchy external key known to the external source. This is a mandatory value. |
| Celebration~220117 | pog_sub_category_key | VARCHAR2(80) | This is the POG sub-category key. This is a POG hierarchy external key known to the external source. This value is mandatory. |
| CLS | assort_product_level | VARCHAR2(80) | This is an identifier to the product level within the product hierarchy. This value must match the ASO product hierarchy. |
| CLS~19~877 | assort_product_key | VARCHAR2(80) | This is an identifier to a node within the merchandise hierarchy. It can be a specific product or any other node not higher than the assortment product category level within the merchandise hierarchy. |
| 100 | demand_spread_factor | NUMBER(6,3) | This is the demand spread factor. This value is normally null, meaning that a 100% demand is assigned to the POG node. In specific cases where the product is placed on multiple POG nodes, a demand spread factor can be used to split the demand across those multiple POGs. Values can be from 0 to 100. |

## Assortment and Space Optimization POG Season to Assortment Mapping File

This file contains the POG season-to-assortment date mapping. Once the mapping from product to POG has been performed, a second pass examines this table to identify the specific correct season for the POG to use, based on the assortment start date.

*Table 6–23 Assortment and Space Optimization POG Season-to-Assortment Mapping File*

| Example | Field | Type | Description |
|---|---|---|---|
| Decoration~22 | pog_dept_key | VARCHAR2(80) | This is the POG dept key. This is a POG hierarchy external key known to the external source. This is a mandatory value. |
| Xmas~2201 | pog_category_key | VARCHAR2(80) | This is the POG category key. This is a POG hierarchy external key known to the external source. This is a mandatory value. If the sub-category key is missing, then this value will become the lowest level within the POG hierarchy. |
| Celebration~220117 | pog_sub_category_key | VARCHAR2(80) | This is the POG sub-category key. This is a POG hierarchy external key known to the external source. This value can be missing. If it is present this is the lowest level within the POG hierarchy |

*Table 6–23 (Cont.) Assortment and Space Optimization POG Season-to-Assortment Mapping File*

| Example | Field | Type | Description |
|---|---|---|---|
| Winter | seasonal_attribute | VARCHAR2(30) | This refers to a specific year-independent time period (season) for a CM assortment and a POG set. Examples include Spring, holiday, back to school, year-round, Fall, Winter. |
| 0000-12-21 | min_assort_start_dt | Date | The year component is irrelevant. The year component should be delivered as 0000. This is a year-independent time period. The assortment start date is matched within the date range specified by this minimum assortment start date and the maximum assortment start date. The format is YYYY-MM-DD. |
| 0000-03-20 | max_assort_start_dt | Date | The year component is irrelevant. The year component should be delivered as 0000.This is a year-independent time period. The assortment start date is matched within the date range specified by the minimum assortment start date and this maximum assortment start date. The format is YYYY-MM-DD. |

### Assortment and Space Optimization Product Stack Height Limit File

This file is used to accept an optional client feed that provides product-specific stacking height limits.

*Table 6–24 Assortment and Space Optimization POG Season-to-Assortment Mapping File*

| Example | Field | Type | Description |
|---|---|---|---|
| 1234582 | product_key | VARCHAR2(80) | This is the external product ID that is known and shared across applications. |
| 22 | stack_height_limit | NUMBER(18,4) | This is the stacking height limit for the specific product. The value must be provided using the same measurement units that are used for all other product dimensions. |
| Y | enabled_flg | VARCHAR2(1) | This flag indicates if the product stacking height limit should be used or not. Y means that the value specified will be used. N means that the value will be ignored and the application global value will be used for the product. |

# ASO Export

ASO generates results for optimized assortments, optimized POGs, and replenishment updates.

## ASO Output Views

ASO does not generate export files. Instead, a set of database views is created representing the output of the ASO application. It is up to the implementation team to extract what is needed from these views and import any required results into CM and optionally a replenishment system such as RMS/AIP. ASO does not support database views for planogram results. These views define ASO's output for CM and AIP (or other replenishment system).

### SO_ASSORT_INT_VW

This is a database view for ASO output to CM.

*Table 6–25 SO_ASSORT_INT_VW*

| Column Name | Data Type | Comments | Nullable |
|---|---|---|---|
| SO_ASSORTMENT_ID | NUMBER(10) | Application internal unique assortment ID. | N |
| LABEL | VARCHAR2(50) | Assortment label as received from the external interface. | Y |
| TRADE_AREA_LABEL | VARCHAR2(80) | Trade area label as received from the external interface. | Y |
| ASSORTMENT_EXT_ID | VARCHAR2(80) | Assortment external ID/key as received from the external interface. | Y |
| ASSORTMENT_SET_ID | VARCHAR2(80) | External value used by APO to bring together multiple user requests that belong to the same assortment group. | Y |
| ASSORT_PRODUCT_ CATEGORY_KEY | VARCHAR2(80) | Assortment product category external key. | N |
| ASSORT_PRODUCT_ CATEGORY_NAME | VARCHAR2(80) | Assortment product category name | N |
| ASSORT_ROLE | VARCHAR2(50) | Assortment role as received from the external interface. | Y |
| ASSORT_TACTIC | VARCHAR2(100) | Assortment tactic as received from the external interface. | Y |
| ASSORT_GOAL | VARCHAR2(50) | Assortment goal as received from the external interface. | Y |
| SO_ASSORT_CLUSTER_ID | NUMBER(10) | Application internal unique assortment cluster ID. | N |
| ASSORT_CLUSTER_NAME | VARCHAR2(80) | Assortment cluster name as received from the external interface. | N |
| EXT_CLUSTER_KEY | VARCHAR2(80) | Assortment cluster external ID or key, as received from the external interface. | N |
| START_DT | DATE | Assortment cluster or store start date as received from the external interface. | Y |
| END_DT | DATE | Assortment cluster or store end date as received from the external interface. | Y |
| SO_LOC_HIER_ID | NUMBER(10) | Application internal unique store ID. | N |
| LOC_EXT_KEY | VARCHAR2(80) | Store external key as received from the external interface. | N |
| LOC_EXT_CODE | VARCHAR2(80) | Store external ID or code as received from the external interface. | Y |
| LOC_NAME | VARCHAR2(255) | Application store internal name. | Y |
| SO_PROD_HIER_ID | NUMBER(10) | Application internal unique product ID. | Y |

*Table 6–25 (Cont.) SO_ASSORT_INT_VW*

| Column Name | Data Type | Comments | Nullable |
|---|---|---|---|
| PROD_EXT_KEY | VARCHAR2(80) | Product external key as received from the external interface. | N |
| PROD_EXT_CODE | VARCHAR2(80) | Product external ID or code as received from the external interface. | Y |
| PROD_NAME | VARCHAR2(255) | Product name. | Y |
| FACING_QTY | NUMBER | Application calculated number of facings. | Y |
| SALES_QTY | NUMBER | Application calculated sales units. | Y |
| SALES_AMT | NUMBER | Application calculated sales amount. | Y |
| MARGIN_AMT | NUMBER | Application calculated margin value. | Y |
| DAYS_OF_SUPPLY | NUMBER | Application-calculated days of supply. | Y |
| LOST_SALES_QTY | NUMBER | Application-calculated lost sales units. | Y |
| DEMAND_QTY | NUMBER | Application-calculated demand units. | Y |
| SERVICE_LEVEL | NUMBER | Application-calculated service level. | Y |
| POGSET_LOCATION_COUNT | NUMBER | POG set location count with facing quantity greater than zero. | Y |
| POG_LENGTH | NUMBER | Main POG length for the SKU/Store | Y |
| CATEGORY_LENGTH | NUMBER | Total POG length across POGs that only include products for the single category | Y |
| EXPORTED_DT | DATE | This date/time is internal to space optimization application and it indicates the time when the user accepts the assortment results and they become available to external applications. | N |

## SO_ASSORT_CM_INT_VW

This is a database view defining ASO output to a replenishment system. (AIP is the template.)

*Table 6–26 SO_ASSORT_CM_INT_VW*

| Column Name | Data Type | Comments | Nullable |
|---|---|---|---|
| SO_ASSORTMENT_ID | NUMBER(10) | Application internal unique assortment ID. | N |
| LABEL | VARCHAR2(50) | Assortment label as received from the external interface. | Y |
| TRADE_AREA_LABEL | VARCHAR2(80) | Trade area label as received from the external interface. | Y |
| ASSORTMENT_SET_ID | VARCHAR2(80) | External value used by APO to bring together multiple user requests that belong to the same assortment Group. | N |

*Table 6–26 (Cont.) SO_ASSORT_CM_INT_VW*

| Column Name | Data Type | Comments | Nullable |
|---|---|---|---|
| ASSORTMENT_EXT_ID | VARCHAR2(80) | Assortment external ID or key as received from the external interface. | Y<br>This value is NULL for the Assortment Finalization records, since multiple assortment_ext_ids are included in those results. |
| ASSORT_PRODUCT_CATEGORY_KEY | VARCHAR2(80) | Assortment product category external key. | N |
| ASSORT_PRODUCT_CATEGORY_NAME | VARCHAR2(80) | Assortment product category name. | N |
| ASSORT_ROLE | VARCHAR2(50) | Assortment role as received from the external interface. | Y |
| ASSORT_TACTIC | VARCHAR2(100) | Assortment tactic as received from the external interface. | Y |
| ASSORT_GOAL | VARCHAR2(50) | Assortment goal as received from the external interface. | Y |
| SO_ASSORT_CLUSTER_ID | NUMBER(10) | Application internal unique assortment cluster ID. | N |
| ASSORT_CLUSTER_NAME | VARCHAR2(80) | Assortment cluster name as received from the external interface. | N |
| EXT_CLUSTER_KEY | VARCHAR2(80) | Assortment cluster external ID or key, as received from the external interface. | N |
| START_DT | DATE | Assortment cluster or store start date as received from the external interface. | Y |
| END_DT | DATE | Assortment cluster or store end date as received from the external interface. | Y |
| SO_LOC_HIER_ID | NUMBER(10) | Application internal unique store ID. | N |
| LOC_EXT_KEY | VARCHAR2(80) | Store external key as received from the external interface. | N |
| LOC_EXT_CODE | VARCHAR2(80) | Store external ID or code as received from the external interface. | Y |
| LOC_NAME | VARCHAR2(255) | Application store internal name. | Y |
| SO_PROD_HIER_ID | NUMBER(10) | Application internal unique product ID. | Y |
| PROD_EXT_KEY | VARCHAR2(80) | Product external key as received from the external interface. | N |
| PROD_EXT_CODE | VARCHAR2(80) | Product external ID or code as received from the external interface. | Y |
| PROD_NAME | VARCHAR2(255) | Product name. | Y |

*Table 6–26 (Cont.) SO_ASSORT_CM_INT_VW*

| Column Name | Data Type | Comments | Nullable |
|---|---|---|---|
| FACING_QTY | NUMBER | Application calculated number of facings. | Y |
| SALES_QTY | NUMBER | Application calculated sales units. | Y |
| SALES_AMT | NUMBER | Application calculated sales amount. | Y |
| MARGIN_AMT | NUMBER | Application calculated margin value. | Y |
| POG_LENGTH | NUMBER | Main POG length for the SKU/store. | Y |
| CATEGORY_LENGTH | NUMBER | Total POG Length across POGs that only include products for the single category. | Y |
| EXPORTED_DT | DATE | This date/time is internal to space optimization application and it indicates the time the user accepts the assortment results and they become available to external applications. | N |

### SO_ASSORT_AIPREPL_INT_VW

This view is a composite view containing contents of both SO_ASSORT_CM_INT_VW and SO_ASSORT_AIPREPL_INT_VW.

*Table 6–27 SO_ASSORT_AIPREPL_INT_VW*

| Column Name | Data Type | Comments | Nullable |
|---|---|---|---|
| SO_ASSORTMENT_ID | NUMBER(10) | Application internal unique assortment ID. | N |
| ASSORTMENT_SET_ID | VARCHAR2(80) | External value used by APO to bring together multiple user requests that belong to the same assortment group. | N |
| ASSORTMENT_EXT_ID | VARCHAR2(80) | Assortment external ID or key as received from the external interface. | Y<br>This value is NULL for the Assortment Finalization records, since multiple assortment_ext_ids are included in those results. |
| SO_LOC_HIER_ID | NUMBER(10) | Application internal unique store ID. | N |
| LOC_EXT_KEY | VARCHAR2(80) | Store external key as received from the external interface. | N |
| LOC_EXT_CODE | VARCHAR2(80) | Store external ID or code as received from the external interface. | Y |
| LOC_NAME | VARCHAR2(255) | Application store internal name. | Y |
| SO_PROD_HIER_ID | NUMBER(10) | Application internal unique product ID. | Y |
| PROD_EXT_KEY | VARCHAR2(80) | Product external key as received from the external interface. | N |

*Table 6–27 (Cont.) SO_ASSORT_AIPREPL_INT_VW*

| Column Name | Data Type | Comments | Nullable |
|---|---|---|---|
| PROD_EXT_CODE | VARCHAR2(80) | Product external ID or code as received from the external interface. | Y |
| PROD_NAME | VARCHAR2(255) | Product name. | Y |
| SERVICE_LEVEL | NUMBER | Application calculated service level. | Y |
| POGSET_LOCATION_COUNT | NUMBER | POG Set location count with facing quantity greater than zero. | Y |
| START_DT | DATE | Assortment cluster or store start date as received from the external interface. | Y |
| END_DT | DATE | Assortment cluster or store end date as received from the external interface. | Y |
| DAYS_OF_SUPPLY | NUMBER | Application calculated days of supply. | Y |
| EXPORTED_DT | DATE | This date/time is internal to space optimization application and it indicates the time when the user accepts the assortment results and they become available to external applications. | N |

### SO_POG_ASSORT_INT_VW

This view delivers the cross reference between planograms and finalized assortments that have products placed on them.

*Table 6–28 SO_POG_ASSORT_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POG_KEY | NUMBER(10) | ASO's internal optimization planogram identifier (Run POG ID). | N |
| ASSORTMENT_CODE | VARCHAR2(80) | Finalized Assortment set identifier as received from the external interface.with APO. | N |
| FINALIZED_DT | TIMESTAMP(6) | Date/time the assortment set finalized by APO. This is the time the data from the assortment becomes final and available through this interface. | N |

### SO_POG_DTL_INT_VW

This view delivers POG header information corresponding to planograms that have been used by finalized assortments.

*Table 6–29 SO_POG_DTL_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POG_KEY | NUMBER(10) | ASO's internal optimization planogram identifier (Run POG ID). | N |
| POG_NAME | VARCHAR2 (80) | Optimization planogram's name | Y |
| POG_DESCRIPTION | VARCHAR2 (200) | Optimization planogram's description. | Y |
| POG_REVISION | NUMBER | Not available.<br>Hard coded value = 1 (as requested by MSM). | N |

*Table 6–29 (Cont.) SO_POG_DTL_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POG_STATUS | CHAR (7) | Not available.<br>Hard coded value = Current as requested by MSM). | N |
| POG_DEPT_CODE | VARCHAR2 (80) | Planogram set department key. | N |
| POG_DEPT | VARCHAR2 (80) | Planogram set department name. | N |
| POG_CAT_CODE | VARCHAR2 (80) | Planogram set category key. | N |
| POG_CAT | VARCHAR2 (80) | Planogram set category name. | N |
| POG_SUBCAT_CODE | VARCHAR2 (80) | Planogram set sub-category key. | N |
| POG_SUBCAT | VARCHAR2 (80) | Planogram set sub-category name. | N |
| SEASON | VARCHAR2 (30) | Planogram set seasonal attribute. | N |
| SEASON_CODE | VARCHAR2 (30) | Planogram set season code. | N |
| LENGTH | NUMBER (18,4) | Planogram length. | N |
| DEPTH | NUMBER (18,4) | Planogram maximum depth. | N |
| HEIGHT | NUMBER (18,4) | Planogram maximum height. | N |
| MODIFIED_DATE | TIMESTAMP(6) | Finalized date. This field hosts the last time an assortment that uses the planogram was finalized. This value should be used to identify changes on the planogram data set. | N |
| EFFECTIVE_DATE | DATE | Planogram start date. | N |
| EXPIRY_DATE | DATE | Planogram end date. | N |
| CAN_SPLIT | NUMBER | Not available.<br>Hard coded value = 0 (as requested by MSM). | N |
| TRAFIC_FLOW_L_TO_R | NUMBER | Traffic flow left to Right.<br>Not available.<br>Hard coded value = 1 (as requested by MSM). | N |
| ASSORTMENT_CODE | VARCHAR2 (20) | Assortment code is not provided here; instead there is a separate cross reference table that provided that data. Many assortments can be linked to the same planogram. | Y |
| CATEGORY_ROLE | VARCHAR2 (20) | Not available. | Y |
| CATEGORY_STRATEGY | VARCHAR2 (20) | Not available. | Y |
| TEXT1 … TEXT20 | VARCHAR2 (20) | Not available. | Y |
| NUMBER1…NUMBER20 | NUMBER | Not available. | Y |
| FLAG1…FLAG10 | VARCHAR2 (1) | Not available. | Y |
| DATE1…DATE5 | DATE | Not available. | Y |
| SOURCE | VARCHAR2 (3) | Hard coded value = ASO (as requested by MSM). | N |

### SO_POG_EQUIPMENT_INT_VW

This view delivers planogram equipment components for planograms that have been used by finalized assortments.

Note that WT indicates wall thickness. Chest walls are calculated as follows: The ASO view assumes that the capacity is central to the chest dimensions in X and Y and at the top in Z. So wall thickness is (chest size - capacity)/2 and the base size is (chest height - capacity).

*Table 6–30 SO_POG_EQUIPMENT_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POG_KEY | VARCHAR2(80) | External planogram identifier. | N |
| EQUIPMENT_KEY | VARCHAR2 (46) | ASO's internal optimization planogram equipment identifier.<br>Shelves IDs are prefixed with "Shelf-" to make them unique and prevent collision with fixture IDs. | N |
| PARENT_EQUIPMENT_KEY | VARCHAR2 (80) | This value is populated only for shelves. The value is a reference to the parent ASO's internal optimization shelf fixture.<br>Shelf fixtures in this interface are denoted as "Base". | N |
| EQUIPMENT_NAME | VARCHAR2 (80) | Equipment (fixture/shelf) name. | N |
| EQUIPMENT_TYPE | VARCHAR2 (320) | Four different equipment types are delivered.<br>Shelf fixtures = "Base"<br>Freezer fixtures = "Freezer Chest"<br>Pegboard fixtures = "Pegboard"<br>Shelves = "Shelf"<br>ASO "Bays" are not included in the interface. | N |
| EQUIPMENT_COLOR | NUMBER | Not available<br>Hard coded value = 13882323 (as requested by MSM). | N |
| LENGTH_SIZE | NUMBER (18,4) | Fixture of shelf length. | N |
| DEPTH_SIZE | NUMBER (18,4) | Fixture of shelf depth. | N |
| HEIGHT_SIZE | NUMBER (18,4) | ASO receives and uses the freezer dimensions rotated as shown below. The view transposes the dimensions to turn the freezer back to the original position.<br>The rotation is performed as follows (freezers only)<br>ASO length remains as length in the view.<br>ASO depth is delivered as height in the view<br>ASO height is delivered as depth in the view. | N |

*Table 6–30 (Cont.) SO_POG_EQUIPMENT_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| HORIZONTAL_POSITION | NUMBER | Horizontal position within the planogram.<br>Each base, pegboard or freezer chest horizontal position is adjusted to account for the prior bays/fixtures within the planogram (The EQUIPMENT view aggregates the positions of the fixtures to allow for the bay sequences.) | N |
| DEPTH_POSITION | NUMBER | Depth shelf/fixture position. | N |
| VERTICAL_POSITION | NUMBER | Vertical shelf/fixture position. | N |
| ROTATION_SLOPE_X | NUMBER | Not available.<br>Hard coded value = 0 (as requested by MSM). | N |
| ROTATION_ROLL_Y | NUMBER | Not available.<br>Hard coded value = 0 (as requested by MSM). | N |
| ROTATION_ANGLE_Z | NUMBER | Not available<br>Hard coded value = 0 (as requested by MSM) | N |
| WT_LEFT | NUMBER | (Chest length - capacity_x) /2 | N |
| WT_RIGHT | NUMBER | (Chest length - capacity_x) /2 | N |
| WT_FRONT | NUMBER | This uses ASO dimensions before rotating the freezer.<br>(Chest height - capacity_z)/2 | N |
| WT_BACK | NUMBER | This uses ASO dimensions before rotating the freezer.<br>(Chest height - capacity_z)/2 | N |
| WT_BASE | NUMBER | This uses ASO dimensions before rotating the freezer.<br>(Chest depth - capacity_y)/ | N |
| WT_TOP | NUMBER | This is set to zero. | N |
| DIVIDERS_WIDTH | NUMBER | Not available. | Y |
| DIVIDERS_DEPTH | NUMBER | Not available. | Y |
| DIVIDERS_HEIGHT | NUMBER | Not available. | Y |
| DIVIDERS_AT_START | NUMBER | Not available.<br>Hard coded value = 0 (as requested by MSM). | N |
| DIVIDERS_AT_END | NUMBER | Not available.<br>Hard coded value = 0 (as requested by MSM). | N |
| DIVIDERS_BETWEEN | NUMBER | Not available.<br>Hard coded value = 0 (as requested by MSM). | N |
| GRILLE_HEIGHT | NUMBER | Not available. | Y |
| PEG_START_HORIZONTAL | NUMBER | Not available. | Y |
| PEG_START_VERTICAL | NUMBER | Not available. | Y |

*Table 6–30 (Cont.) SO_POG_EQUIPMENT_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PEG_WIDTH | NUMBER | Horizontal spacing. | Y |
| PEG_HEIGHT | NUMBER | Vertical spacing. | Y |
| PEG_DROP | NUMBER | Not available. | Y |
| FINALIZED_DT | TIMESTAMP(6) | Date/time the assortment set was finalized by APO. This is the time the data from the assortment becomes final and available through this interface. | N |

### SO_POG_LOC_INT_VW

This view delivers planogram/stores cross reference (for planograms that have been used by finalized assortments)

*Table 6–31 SO_POG_LOC_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POG_KEY | NUMBER (10) | ASO's internal optimization planogram identifier (Run POG ID) | N |
| STORE_KEY | VARCHAR2 (80 Char) | This is the external store ID, known and shared across applications (RMS ID) | N |
| EFFECTIVE_DATE | DATE | Planogram start date | N |
| EXPIRY_DATE | DATE | Planogram end date | N |
| SALES | NUMBER | Not available | Y |
| PROFIT | NUMBER | Not available | Y |
| MOVEMENT | NUMBER | Not available | Y |
| TEXT1 | VARCHAR2 (80 Char) | Optimization level string attribute 1 | Y |
| TEXT2 | VARCHAR2 (80 Char) | Optimization level string attribute 2 | Y |
| TEXT3 | VARCHAR2 (80 Char) | Optimization level string attribute 3 | Y |
| TEXT4 | VARCHAR2 (80 Char) | Optimization level string attribute 4 | Y |
| TEXT5 | VARCHAR2 (80 Char) | Optimization level string attribute 5 | Y |
| TEXT6 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT7 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT8 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT9 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT10 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT11 | VARCHAR2 (20 Char) | Not available | Y |

*Table 6–31 (Cont.) SO_POG_LOC_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| TEXT12 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT13 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT14 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT15 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT16 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT17 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT18 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT19 | VARCHAR2 (20 Char) | Not available | Y |
| TEXT20 | VARCHAR2 (20 Char) | Not available | Y |
| NUMBER1 | NUMBER (18,4) | Optimization level numeric attribute 1 | Y |
| NUMBER2 | NUMBER (18,4) | Optimization level numeric attribute 2 | Y |
| NUMBER3 | NUMBER (18,4) | Optimization level numeric attribute 3 | Y |
| NUMBER4 | NUMBER (18,4) | Optimization level numeric attribute 4 | Y |
| NUMBER5 | NUMBER (18,4) | Optimization level numeric attribute 5 | Y |
| NUMBER6 | NUMBER (18,4) | Optimization level numeric attribute 6 | Y |
| NUMBER7 | NUMBER (18,4) | Optimization level numeric attribute 7 | Y |
| NUMBER8 | NUMBER (18,4) | Optimization level numeric attribute 8 | Y |
| NUMBER9 | NUMBER (18,4) | Optimization level numeric attribute 9 | Y |
| NUMBER10 | NUMBER (18,4) | Optimization level numeric attribute 10 | Y |
| NUMBER11 | NUMBER (5,4) | Optimization level percentage attribute 1 | Y |
| NUMBER12 | NUMBER (5,4) | Optimization level percentage attribute 2 | Y |
| NUMBER13 | NUMBER (5,4) | Optimization level percentage attribute 3 | Y |
| NUMBER14 | NUMBER (5,4) | Optimization level percentage attribute 4 | Y |
| NUMBER15 | NUMBER (5,4) | Optimization level percentage attribute 5 | Y |
| NUMBER16 | NUMBER | Not available | Y |
| NUMBER17 | NUMBER | Not available | Y |
| NUMBER18 | NUMBER | Not available | Y |
| NUMBER19 | NUMBER | Not available | Y |
| NUMBER20 | NUMBER | Not available | Y |
| FLAG1 | VARCHAR2 (1 Byte) | Not available | Y |
| FLAG2 | VARCHAR2 (1 Byte) | Not available | Y |

*Table 6–31 (Cont.) SO_POG_LOC_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| FLAG3 | VARCHAR2 (1 Byte) | Not available | Y |
| FLAG4 | VARCHAR2 (1 Byte) | Not available | Y |
| FLAG5 | VARCHAR2 (1 Byte) | Not available | Y |
| FLAG6 | VARCHAR2 (1 Byte) | Not available | Y |
| FLAG7 | VARCHAR2 (1 Byte) | Not available | Y |
| FLAG8 | VARCHAR2 (1 Byte) | Not available | Y |
| FLAG9 | VARCHAR2 (1 Byte) | Not available | Y |
| FLAG10 | VARCHAR2 (1 Byte) | Not available | Y |
| DATE1 | DATE | Optimization level date attribute 1 | Y |
| DATE2 | DATE | Optimization level date attribute 2 | Y |
| DATE3 | DATE | Optimization level date attribute 3 | Y |
| DATE4 | DATE | Optimization level date attribute 4 | Y |
| DATE5 | DATE | Optimization level date attribute 5 | Y |
| FINALIZED_DT | TIMESTAMP(6) | Date/time the assortment set was finalized by APO. This is the time the data from the assortment becomes final and available through this interface. | N |

## SO_POG_PROD_HIER_INT_VW

This view delivers finalized assortment products hierarchies. ASO uses generic product hierarchy levels. (Each installation could have different names and levels.) MSM requested the following names for the given hierarchy levels: Level 1: Company, Level 2: Division, Level 3: Group, Level 4: Department, Level 5: Class, Level 6: Subclass, Level 7: Style.

*Table 6–32 SO_POG_PROD_HIER_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| COMPANY | VARCHAR2 (255) | ASO hierarchy level 1 (node) name | N |
| COMPANY_CODE | VARCHAR2 (80) | ASO hierarchy level 1. This is the external hierarchy level ID that is known and shared across applications. (From RMS) | N |
| DIVISION | VARCHAR2 (255) | ASO hierarchy level 2 (node) name | N |
| DIVISION_CODE | VARCHAR2 (80) | ASO hierarchy level 2. This is the external hierarchy level ID that is known and shared across applications. (From RMS) | N |
| GRP | VARCHAR2 (255) | ASO hierarchy level 13 (node) name | N |

*Table 6–32 (Cont.) SO_POG_PROD_HIER_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| GRP_CODE | VARCHAR2 (80) | ASO hierarchy level 3. This is the external hierarchy level ID that is known and shared across applications. (From RMS) | N |
| DEPARTMENT | VARCHAR2 (255) | SO's hierarchy level 4 (node's) name | N |
| DEPARTMENT_CODE | VARCHAR2 (80) | ASO hierarchy level 4. This is the external hierarchy level ID that is known and shared across applications. (From RMS) | N |
| CLASS | VARCHAR2 (255) | ASO hierarchy level 5 (node) name | N |
| CLASS_CODE | VARCHAR2 (80) | ASO hierarchy level 5. This is the external hierarchy level ID that is known and shared across applications. (From RMS) | N |
| SUBCLASS | VARCHAR2 (255) | ASO hierarchy level 6 (node) name | N |
| SUBCLASS_CODE | VARCHAR2 (80) | ASO hierarchy level 6. This is the external hierarchy level ID that is known and shared across applications.(From RMS) | N |
| STYLE | VARCHAR2 (255) | ASO hierarchy level 7 (node) name | Y |
| STYLE_CODE | VARCHAR2 (80) | ASO hierarchy level 7. This is the external hierarchy level ID that is known and shared across applications. (From RMS) | Y |
| STYLE_COLOR | VARCHAR2 (1) | Not available | Y |
| STYLE_COLOR_CODE | VARCHAR2 (1) | Not available | Y |
| FINALIZED_DT | TIMESTAMP(6) | Date/time the assortment set was finalized by APO. This is the time the data from the assortment becomes final and available through this interface. | N |

## SO_POG_PROD_POS_INT_VW

This view delivers finalized assortment products placed within optimized planograms.

*Table 6–33 SO_POG_PROD_POS_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POG_KEY | NUMBER | ASO internal optimization planogram identifier (Run POG ID) | N |
| EQUIPMENT_KEY | VARCHAR2 (46) | ASO internal optimization planogram equipment identifier. Shelves IDs are prefixed with "Shelf-" to make them unique and prevent collision with fixture IDs.<br>Products can be placed on Pegboards, Freezer chest and shelves (never in a "base"). | N |
| SKU_KEY | VARCHAR2 (80) | This is the external product ID that is known and shared across applications. (From RMS) | N |
| DISPLAY_STYLE_TYPE | VARCHAR2 (80) | Display style type used to place the product in the planogram. For example, Single or Unit, Tray, Case, or Pallet. | N |
| FACING_HORIZONTAL | NUMBER | Number of horizontal facings of the product | N |
| FACING_DEPTH | NUMBER | Number of facings deep | N |

*Table 6–33 (Cont.) SO_POG_PROD_POS_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| FACING_STACK | NUMBER | Number of facings above | N |
| SIZE_LENGTH | NUMBER (18,4) | Facing length dimension | N |
| SIZE_DEPTH | NUMBER (18,4) | Facing depth dimension | N |
| SIZE_HEIGHT | NUMBER (18,4) | Facing height dimension | N |
| HORIZONTAL_POSITION | NUMBER | Product horizontal position relative to the fixture or shelf | N |
| DEPTH_POSITION | NUMBER | Z and Y position of products on shelves are calculated by the view based on the product size and shelf thickness to give the bottom-back-left corner's position of the product relative to the bottom-back-left corner of the shelf, pegboard or chest.<br>The depth product position is calculated like this ((SO_RUN_POG_SHELF.DEPTH - SO_DISPLAY_STYLE.DEPTH)*-1). For a product within freezer a different approach is used to account for the rotation of the axis: (((ASO freezer.height-) (ASO product height in the freezer)-(product_height))*(-1) | N |
| VERTICAL_POSITION | NUMBER | Shelf and freezer are set to zero.<br>Pegboards The Z position on pegboards corresponds to the left bottom corner of the product as positioned in the pegboard. | N |
| ROTATION_SLOPE_X | NUMBER | Hard coded value = 0 (as requested by MSM) | Y |
| ROTATION_ROLL_Y | NUMBER | Hard coded value = 0 (as requested by MSM) | Y |
| ROTATION_ANGLE_Z | NUMBER | Hard coded value = 0 (as requested by MSM) | Y |
| ORIENTATION | VARCHAR2 (80 Char) | String value corresponding to the orientation used to place the product. For example, Front 0, Top 90, Bottom 180. | Y |
| VC_ORIENTATION | VARCHAR2 (80 Char) | VC = Vertical Caps<br>Not available | Y |
| VC_FACING_HORIZONTAL | NUMBER | Not available | Y |
| VC_FACING_DEPTH | NUMBER | Not available | Y |
| VC_FACING_STACKING | NUMBER | Not available | Y |
| VC_SIZE_LENGTH | NUMBER | Not available | Y |
| VC_SIZE_DEPTH | NUMBER | Not available | Y |
| VC_SIZE_HEIGHT | NUMBER | Not available | Y |
| SC_ORIENTATION | NUMBER | SC=Side Caps<br>Not available | Y |
| SC_FACING_HORIZONTAL | NUMBER | Not available | Y |
| SC_FACING_DEPTH | NUMBER | Not available | Y |
| SC_FACING_STACKING | NUMBER | Not available | Y |
| SC_SIZE_LENGTH | NUMBER | Not available | Y |

*Table 6–33 (Cont.) SO_POG_PROD_POS_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| SC_SIZE_DEPTH | NUMBER | Not available | Y |
| SC_SIZE_HEIGHT | NUMBER | Not available | Y |
| FINALIZED_DT | TIMESTAMP(6) | Date/time the assortment set was finalized by APO. This is the time the data from the assortment becomes final and available through this interface. | Y |

### SO_POG_SKU_DTL_INT_VW

This view contains product's display style information. It calculates some of the metrics under the assumption that each product has a unit display style defined.

*Table 6–34 SO_POG_SKU_DTL_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| SKU_KEY | VARCHAR2 (80) | This is the external product ID that is known and shared across applications. (From RMS) | N |
| SKU_NAME | VARCHAR2 (255) | ASO product description. If NULL then use product's name. | N |
| SKU_ABBREV | VARCHAR2 (255) | ASO product name | N |
| BRAND | VARCHAR2 (1) | Not available | Y |
| MANUFACTURER | VARCHAR2 (1) | Not available | Y |
| SUPPLIER | VARCHAR2 (1) | Not available | Y |
| PARENT_PRODUCT_CODE | VARCHAR2 (80) | This is the external product hierarchy level ID that is known and shared across applications. (From RMS)<br>This value links to a parent within the product hierarchy view. | N |
| EFFECTIVE_DATE | DATE | Not available<br>Set to sysdate as requested by MSM. | N |
| EXPIRY_DATE | DATE | Not available | Y |
| QUANTITY | NUMBER (10) | Units to display from the Units display style | Y |
| COLOR | NUMBER | Not available<br>Set to 16777215 as requested by MSM. | Y |
| SHAPE | CHAR (3) | Not available<br>Set to Box as requested by MSM. | Y |
| DFLT_ORIENTATION | VARCHAR2 (80) | Default orientation from the product's UNIT display style | N |
| VALID_ORIENTATIONS | NUMBER | This value is calculated by adding the bitwise values corresponding to all the orientations that are valid for the product. | N |
| WEIGHT | NUMBER | Not Available | Y |
| LENGTH | NUMBER (18,4) | Product's default unit display style length | N |
| DEPTH | NUMBER (18,4) | Product's default unit display style depth | N |
| HEIGHT | NUMBER (18,4) | Product's default unit display style height | N |

*Table 6–34 (Cont.) SO_POG_SKU_DTL_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| NESTING_HORIZONTAL | NUMBER | Product's default unit display style nesting length | N |
| NESTING_DEPTH | NUMBER | Product's default unit display style nesting depth | N |
| NESTING_VERTICAL | NUMBER | Product's default unit display style nesting height | N |
| HORIZONTAL_CF | NUMBER | CF=Crush Factor<br>Not Available<br>Set to 0 as requested by MSM. | N |
| DEPTH_CF | NUMBER | Not Available<br>Set to 0 as requested by MSM | N |
| VERTICAL_CF | NUMBER | Not Available<br>Set to 0 as requested by MSM | N |
| FINGER_SPACE_HORIZONTAL | NUMBER | Product's default unit display style finger space horizontal | N |
| FINGER_SPACE_DEPTH | NUMBER | Product's default unit display style finger space depth | N |
| FINGER_SPACE_VERTICAL | NUMBER | Product's default unit display style finger space vertical | N |
| MAX_STACKING | NUMBER | Product's default unit display style max stacking | Y |
| JUMBLE_FACTOR | NUMBER | Not Available<br>Set to 0 as requested by MSM. | Y |
| PH1_OFFSET_HORIZONTAL | NUMBER | PH1 = Peg Hole 1<br>Not Available | Y |
| PH1_OFFSET_DEPTH | NUMBER | Not Available | Y |
| PH1_OFFSET_VERTICAL | NUMBER | Not Available | Y |
| PH2_OFFSET_HORIZONTAL | NUMBER | PH2 = Peg Hole 2<br>Not Available | Y |
| PH2_OFFSET_DEPTH | NUMBER | Not Available | Y |
| PH2_OFFSET_VERTICAL | NUMBER | Not Available | Y |
| TRAY_LENGTH | NUMBER | Tray display style length | Y |
| TRAY_DEPTH | NUMBER | Tray display style depth | Y |
| TRAY_HEIGHT | NUMBER | Tray display style height | Y |
| TRAY_QTY_WIDE | NUMBER | Tray display style length/unit display style length | Y |
| TRAY_QTY_DEEP | NUMBER | Tray display style depth/unit display style depth | Y |
| TRAY_QTY_HIGH | NUMBER | Tray display style height/unit display style height | Y |
| CASE_LENGTH | NUMBER | Case display style length | Y |
| CASE_DEPTH | NUMBER | Case display style depth | Y |

*Table 6–34 (Cont.) SO_POG_SKU_DTL_INT_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| CASE_HEIGHT | NUMBER | Case display style height | Y |
| CASE_QTY_WIDE | NUMBER | Case display style length/unit display style length | Y |
| CASE_QTY_DEEP | NUMBER | Case display style depth/unit display style depth | Y |
| CASE_QTY_HIGH | NUMBER | Case display style height/unit display style height | Y |
| DISPLAY_DIM_LENGTH | NUMBER (18,4) | Product's default unit display style length | Y |
| DISPLAY_DIM_DEPTH | NUMBER (18,4) | Product's default unit display style depth | Y |
| DISPLAY_DIM_HEIGHT | NUMBER (18,4) | Product's default unit display style height | Y |
| TEXT1 … TEXT20 | VARCHAR2 (20 Char) | Not available | Y |
| NUMBER1…NUMBER20 | NUMBER | Not available | Y |
| FLAG1…FLAG10 | VARCHAR2 (1 Char) | Not available | Y |
| DATE1…DATE5 | DATE | Not available | Y |
| FINALIZED_DT | TIMESTAMP(6) | Date/time the assortment set was finalized by APO. This is the time the data from the assortment becomes final and available through this interface. | N |

# 7 Market Basket Analysis Overview

This chapter describes Market Basket Analysis (MBA) and contains the following sections:

- Market Basket Functional Overview
- Market Basket Data Mart Backup
- Reclassification Impact
- Market Basket Operations

## Market Basket Functional Overview

Market Basket Analysis is a data mining technique that outputs correlations between various items in a customer's basket.

Market Basket Analysis reports are used to understand what sells with what, and includes the probability and profitability of market baskets. Such a report can be used to plan promotions, optimize product placement, and support store planogram decisions. These reports help you understand the statistical relationship between sales for different merchandise.

See the *Oracle Retail Analytics User Guide* for the Market Basket Reporting Overview.

## Market Basket Data Mart Backup

Market Basket Analysis maintains a history of data mining results for a defined number of weeks. The number of weeks can be specified in the data mining configuration table W_RTL_DMS_CONFIG_G. For configuration details, see the *Oracle Retail Analytics Installation Guide*.

The MBA PLP programs are used to control the number of weeks in MBA tables. Once the number of weeks in those tables reaches the number that is specified in the configuration table, MBA PLP programs drop the earliest partition to keep the number of weeks in the target table from exceeding the maximum value.

Since this partition dropping is done automatically, you should back up all MBA tables on a weekly basis. Tables with partitions that are automatically dropped include:

- W_RTL_MB_SLS_CL_WK_A
- W_RTL_MB_SLS_DP_WK_A
- W_RTL_MB_SLS_SC_WK_A
- W_RTL_MB_SLS_SC_WK_SUP_A

- W_RTL_MB_SLSPR_CUST_SC_WK_A
- W_RTL_MB_SLSPR_SC_WK_A
- W_RTL_MB_SLSPR_SC_WK_SUP_A

## Reclassification Impact

The data mining process is performed on a weekly basis. The sales data is used as source data for the mining process for one week. This can improve performance for the data mining process and data mining ETL loading. However, if any product or organization reclassification occurs during the week, then the weekly mining may not have an accurate result. When a reclassification happens, the data mining process must use the sales data from the specified number of weeks instead of the current one week to get consistent and accurate results.

The RA_PROD_WEEKLY_RECLASS_IND and RA_ORG_WEEKLY_RECLASS_IND parameters indicate if there is any reclassification in the current week. The values of these two parameters are updated by the item and location daily ETL programs and are stored in the RA_C_ODI_PARAM table. These two parameters should not be updated by the batch user manually. In the mining configuration table (W_RTL_DMS_CONFIG_G), the restart history indicator for different mining (such as TOP10_RESTART_HIST_IND) and the mining start date wid for different mining (such as TOP10_MIN_DY_WID) are updated based on these parameters by the mining refresh program.

## Market Basket Operations

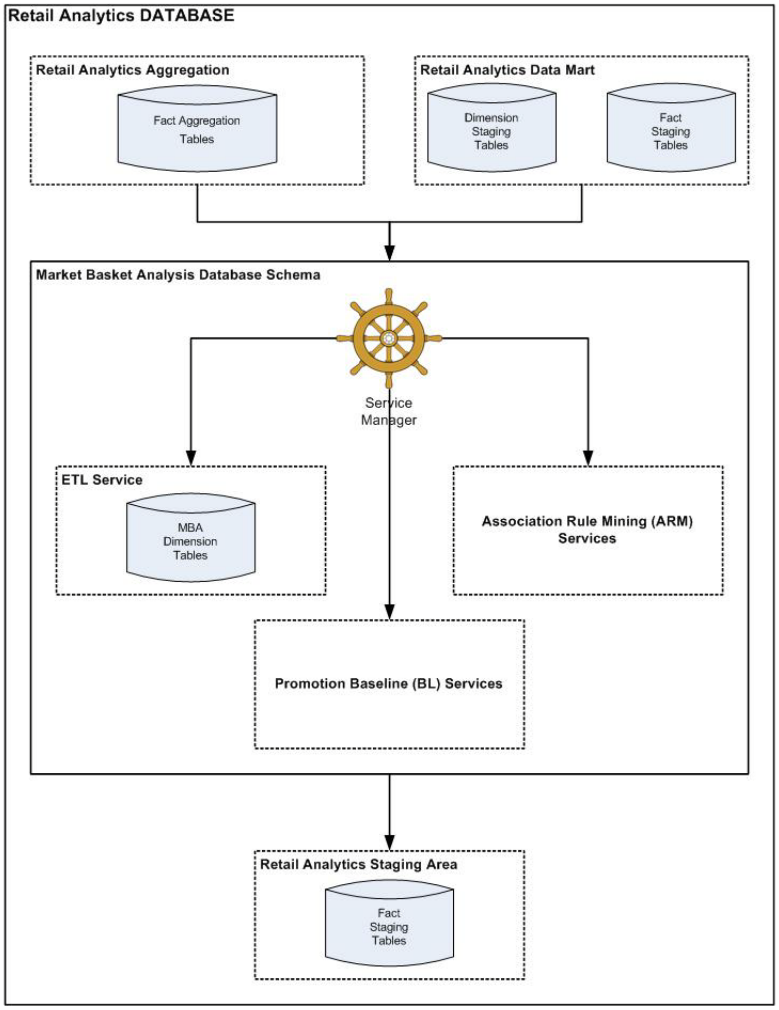
Market Basket Analysis (MBA) helps you to find the relationship between items and groups of items in the basket of a customer. You can also use it to calculate a promotion-based historical baseline in order to provide insight into retail sale patterns and to improve your understanding of promotional effectiveness.

### Market Basket Analysis Architecture

This section outlines the Market Basket Analysis architecture and its components. Review of the architecture is important in understanding the data flow.

Figure 7–1 shows the technical architecture of Market Basket Analysis.

*Figure 7–1 Market Basket Analysis Technical Architecture*


Retail Analytics DATABASE
Retail Analytics Aggregation
Fact Aggregation Tables
Retail Analytics Data Mart
Dimension Staging Tables
Fact Staging Tables
Market Basket Analysis Database Schema
Service Manager
ETL Service
MBA Dimension Tables
Association Rule Mining (ARM) Services
Promotion Baseline (BL) Services
Retail Analytics Staging Area
Fact Staging Tables

Market Basket Analysis consists of the following four components, which are discussed in the subsequent sections.

- Service Manager
- ETL Service
- ARM Services
- Baseline Services

## Extract, Transform, Load

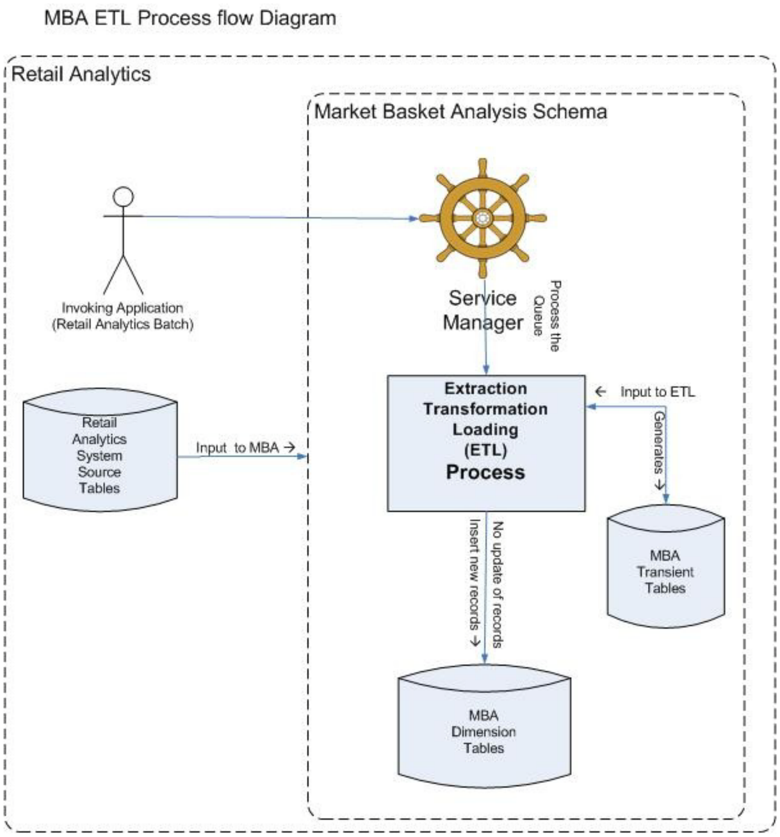
The ETL Service transforms the tables into different organized tables of MBA for efficient data mining.

### Process Flow

MBA resides in a separate set of database schemas and uses ETL scripts to move input data from Retail Analytics into the MBA tables.

Figure 7–2 shows the ETL process.

*Figure 7–2 Market Basket Analysis ETL Process Flow*



ETL programs are added to transform Retail Analytics tables into different organized tables for organization and merchandise hierarchy, customer segments, and calendar for efficient lookups or aggregations for data mining.

The ETL scripts may be run weekly or during a scheduled time frame to refresh the data in the MBA schema before running reports as part of the weekly batch process. The results of any reports are prepared for consumption in Retail Analytics by a separate outbound ETL process that publishes results to predetermined output tables and materialized views, and cleans up any intermediate data in the MBA tables.

The ETL process should be run prior to running any other services to gather information for the MBA required tables. The following routines perform this task:

To set up the ETL service in the queue:

```
begin
  rse_srvc_mgr.batch_pre_process('CORE_ETL','MBA');
end;
```

To process the queue:

```
begin
  rse_srvc_mgr.process_queue('CORE_ETL');
end;
```

A review of the target tables of the ETL program list may be done to validate the ETL data.

### ETL Objects List

Table 7–1 lists the ETL objects.

*Table 7–1 ETL Objects List*

| Program Name | Description | Source Table or File | Target Table |
|---|---|---|---|
| RSE_CORE_ETL. load_merch | Loading of the Merchandising data | W_RTL_DMS_PRODUCT_G | RSE_PROD_TC |
| RSE_CORE_ETL. load_org | Loading of the Organization data | W_RTL_DMS_ORG_DH_G | RSE_INT_ORG_TC |
| RSE_CORE_ETL. load_cur_cust_seg | Loading of the Customer Segment data | W_RTL_CUSTSEG_D | RSE_CUR_CUSTSEG_D |
| RSE_CORE_ETL. load_cur_cust_seg_xref | Loading the current classification of customers to customer segments | W_RTL_CUST_CUSTSEG_D | RSE_CUR_CUST_CUSTSEG_D |
| RSE_CORE_ETL. load_cal | Loading of the calendar data from RA | W_MCAL_DAY_D<br>W_MCAL_WEEK_D | RSE_MCAL_DAY_WK_XREF (Materialized View) |

## ARM Services

ARM Services implements the mining services for Top 10 Product Affinities, Anchor Subclass Top Affinities, Anchor Customer Segment Promotion Affinities, Anchor Subclass Top Affinities by Promotions and Top 10 Promoted Subclass Affinities in order to find the desired association rules.

There are five services associated with ARM, which are discussed in the following sections:

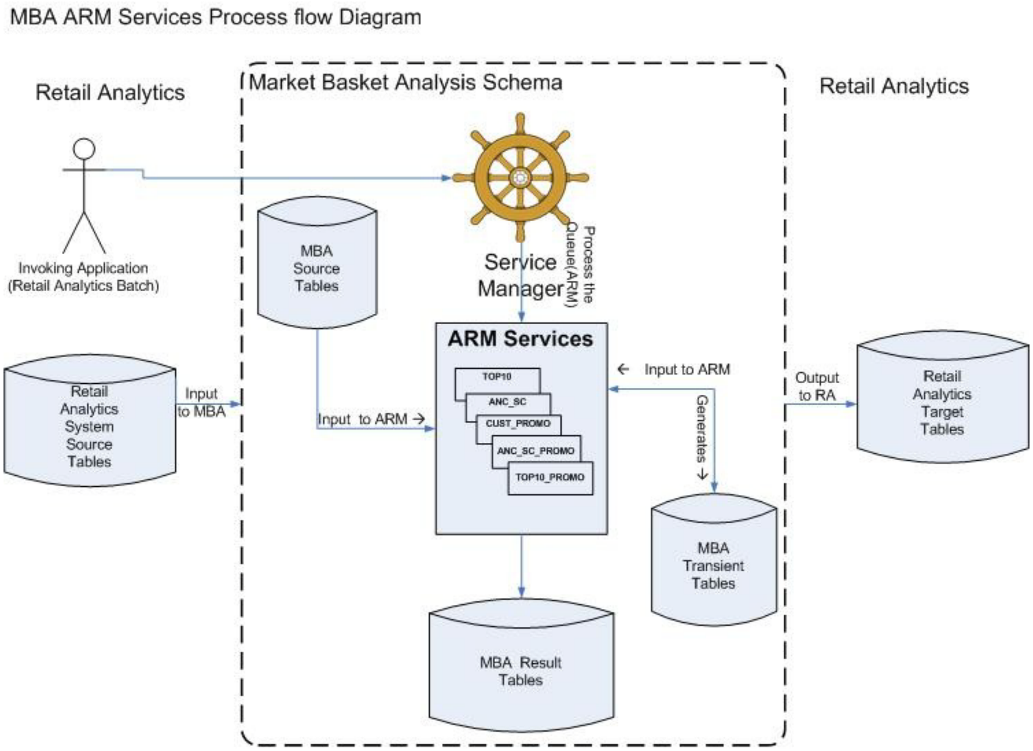
- Top 10 Product Affinities (TOP10)
- Anchor Subclass Top Affinities (ANC_SC)
- Anchor Customer Segment Promotion Affinities (CUST_PROMO)
- Anchor Subclass Top Affinities by Promotions (ANC_SC_PROMO)
- Top 10 Promoted Subclass Affinities (TOP10_PROMO)

Note that since TOP10 and ANC_SC use Oracle Data Mining (ODM), you must obtain ODM licenses prior to executing them in order to comply with the terms of the Oracle Licensing Policy. The CUST_PROMO, ANC_SC_PROMO and TOP10_PROMO do not use ODM.

### Process Flow

Figure 7–3 shows the process flow diagram for the ARM Services.

*Figure 7–3 Market Basket Analysis ARM Services Process Flow*



## ARM Generic Configuration Parameters

Table 7–2 lists the generic parameters that are used for all of the ARM services. They are configured in the table W_RTL_DMS_CONFIG_G, which is populated by an ETL process. The ARM_BATCH_DOP, ARM_WEEKLY_DOP, and ARM_HIST_NUM_OF_WEEK configurations should be determined during initial setup and may be adjusted for performance reasons.

To gain maximum performance benefit, consider configuring the parameters associated with parallel processing.

*Table 7–2 ARM Generic Configuration Parameters*

| Parameter | Type | Configurable | Sample Data | Description |
|---|---|---|---|---|
| ARM_BATCH_DOP | NUMBER | Y | 1 | This is a degree of ARM data mining batches that can be simultaneously processed within each week. |
| ARM_HIST_NUM_OF_WEEK | NUMBER | Y | 16 | This is the number of weeks that the mining result history are held for. |

*Table 7–2 (Cont.) ARM Generic Configuration Parameters*

| Parameter | Type | Configurable | Sample Data | Description |
|---|---|---|---|---|
| ARM_WEEKLY_DOP | NUMBER | Y | 1 | This is a degree of weeks that can be simultaneously processed for ARM data mining when there is more than one week to be processed. This configuration should only be used if the database server is large enough to accommodate it. |
| ARM_WEEK_WID | NUMBER | Y | 124020110038 | This is the current business week number that the mining is executed for. |
| ARM_MIN_SUPPORT_ TXN_CNT | NUMBER | Y | 1000 | This optional configuration provides a lower boundary for the minimum supporting transaction count for any of the affinity processes. This configuration prevents situations where the percentage based value in the other minimum confidence configurations results is too low of a value, if the transaction count fluctuates a lot. |

### Top 10 Product Affinities (TOP10)

The mining service identified as TOP10 is executed against all transactions at either subclass level, all class level, or all department level, based on the system options. The "IF" column can have multiple values up to 3. Navigation to a lower level against the "IF" column is available. Organization hierarchy and rollup are available from this mining output.

To set up the process queue:

```
begin
  rse_srvc_mgr.batch_pre_process('ARM', 'TOP10');
end;
```

**TOP10 Configuration Parameters** The following parameters, listed in Table 7–3, are used for the TOP10 service. They are configured in the table W_RTL_DMS_CONFIG_G, which is populated by an ETL process and can be modified if necessary.

For assistance in setting the configuration parameters for the TOP 10 Product Affinities, refer to the Oracle Retail Analytics Association Rule Mining of Market Basket Data for Retail Analytics Market Basket Analytics White paper (My Oracle Support Note #1469143.1) for guidance on use of a configuration tool. This tool tests various configurations with customer supplied data, in order to arrive at results which satisfy the customers reporting needs.

*Table 7–3 Top 10 Service Configuration Parameters*

| Parameter | Type | Configurable | Sample Data | Description |
|---|---|---|---|---|
| TOP10_MAX_DY_WID | Number | N | 124020100214 000 | This is the last date of data that Top 10 Product Affinities mining program looks at. In most cases, it is the last date of the current week. |
| TOP10_MAX_SET_ SIZE | NUMBER | Y | 4 | This is the total number of items in the Rule for Top10 Product Affinities. |
| TOP10_MIN_ CONFIDENCE_CLS | NUMBER | Y | 0.05 | This is the minimum confidence filter for affinities calculated at the Class level. |

*Table 7–3 (Cont.) Top 10 Service Configuration Parameters*

| Parameter | Type | Configurable | Sample Data | Description |
|---|---|---|---|---|
| TOP10_MIN_CONFIDENCE_DEPT | NUMBER | Y | 0.05 | This is the minimum confidence filter for affinities calculated at the Department level. |
| TOP10_MIN_CONFIDENCE_SBC | NUMBER | Y | 0.05 | This is the minimum confidence filter when Top 10 Product Affinities is configured at the Subclass level. |
| TOP10_MIN_DY_WID | NUMBER | N | 124020100214000 | This is the first date of data that Top 10 Product Affinities mining program looks at. In most cases, it is the first date of current week. This is reset to the first day of the week, that is, the number of weeks that is defined in the parameter ARM_HIST_NUM_OF_WEEK, when there is a reclassification on the product hierarchy in the current week. |
| TOP10_MIN_REVERSE_CONFIDENCE_CLS | NUMBER | Y | 0 | This is the minimum reverse confidence filter for affinities calculated at the Class level. |
| TOP10_MIN_REVERSE_CONFIDENCE_DEPT | NUMBER | Y | 0 | This is the minimum reverse confidence filter for affinities calculated at the Department level. |
| TOP10_MIN_REVERSE_CONFIDENCE_SBC | NUMBER | Y | 0 | This is the minimum reverse confidence filter when Top 10 Product Affinities is configured at the Subclass level. |
| TOP10_MIN_SUPPORT_CLS | NUMBER | Y | 0.0005 | This is the minimum support filter for affinities calculated at the Class level. |
| TOP10_MIN_SUPPORT_DEPT | NUMBER | Y | 0.0005 | This is the minimum support filter for affinities calculated at the Department level. |
| TOP10_MIN_SUPPORT_SBC | NUMBER | Y | 0.0005 | This is the minimum support filter when Top 10 Product Affinities is configured at subclass level. |
| TOP10_PROD_HIER_LEVEL | VARCHAR2 | Y | DEPT | This is the highest product hierarchy level at which the mining for Top 10 Product Affinities is executed. Valid value in ('SBC', 'CLS', 'DEPT'). |
| TOP10_RESTART_HIST_IND | VARCHAR2 | N | N | Valid value in ('Y', 'N'). |

*Table 7–3 (Cont.) Top 10 Service Configuration Parameters*

| Parameter | Type | Configurable | Sample Data | Description |
|---|---|---|---|---|
| TOP10_ MIN_ SUPPORT_TXN_CNT_ DEPT | NUMBER | Y | 1000 | This optional configuration overrides the ARM_MIN_SUPPORT_TXN_CNT configuration and provides a lower boundary for the minimum supporting transaction count for the Department level reporting. |
| TOP10_ MIN_ SUPPORT_TXN_CNT_ CLS | NUMBER | Y | 1000 | This optional configuration overrides the ARM_MIN_SUPPORT_TXN_CNT configuration and provides a lower boundary for the minimum supporting transaction count for the Class level reporting. |
| TOP10_ MIN_ SUPPORT_TXN_CNT_ SBC | NUMBER | Y | 1000 | This optional configuration overrides the ARM_MIN_SUPPORT_TXN_CNT configuration and provides a lower boundary for the minimum supporting transaction count for the Subclass level reporting. |

### Anchor Subclass Top Affinities (ANC_SC)

This mining service is identified as ANC_SC. Subclasses that are used for mining are stored in the configuration table W_RTL_DMS_ATTR_LIST_G. Only subclasses that are found in this table are included in the output of this process. The "IF" column can have multiple values up to two focused subclasses and one excluded subclass. The list of excluded subclasses is limited to those that also have been found to have associations. Organization hierarchy and rollup are not available from this mining output.

To set up the process queue:

```
begin
  rse_srvc_mgr.batch_pre_process('ARM', 'ANC_SC');
end;
```

**ANC_SC Configuration Parameters** The following parameters, listed in Table 7–4, are used for the ANS_SC service. They are configured in the table W_RTL_DMS_CONFIG_G, which is populated by an ETL process and can be modified if necessary.

*Table 7–4 ANS_SC Service Configuration Parameters*

| Parameter | Type | Configurable | Sample Data | Description |
|---|---|---|---|---|
| ANC_SC_ATTR_LIST_ SBC | VARCHAR2 | N | ANC_SC_ ATTR_LIST_ SBC_NUM | This is an identifier of the subclass attribute list to be processed from W_ RTL_DMS_ATTR_LIST_G. |
| ANC_SC_IF_HIER_ LEVEL | VARCHAR2 | N | SBC | This is the product hierarchy level for IF item for the mining for Anchor Subclass Top Affinities. |
| ANC_SC_MAX_DY_ WID | NUMBER | N | 124020100214 000 | This is the last date of data that Anchor Subclass Top Affinities mining program looks at. In most cases, it is the last date of the current week. |
| ANC_SC_MAX_SET_ SIZE | NUMBER | Y | 4 | This is the total number of items in the Rule for Anchor Subclass Top Affinities. |
| ANC_SC_MIN_ CONFIDENCE | NUMBER | Y | 0.05 | This is the minimum confidence filter for Anchor Subclass Top Affinities. |

*Table 7–4 (Cont.) ANS_SC Service Configuration Parameters*

| Parameter | Type | Configurable | Sample Data | Description |
|---|---|---|---|---|
| ANC_SC_MIN_DY_WID | NUMBER | N | 124020100214000 | This is the first date of data that Anchor Subclass Top Affinities mining program looks at. In most cases, it is the first date of current week. This is reset to the first day of the week, that is, the number of weeks ago that is defined in the parameter ARM_HIST_NUM_OF_WEEK, when there is a reclassification on product hierarchy in the current week. |
| ANC_SC_MIN_REVERSE_CONFIDENCE | NUMBER | Y | 0 | This is the minimum reverse confidence filter for Anchor Subclass Top Affinities. |
| ANC_SC_MIN_SUPPORT | NUMBER | Y | 0.0005 | This is the minimum support filter for Anchor Subclass Top Affinities. |
| ANC_SC_RESTART_HIST_IND | VARCHAR2 | N | N | Valid value in ('Y', 'N'). |
| ANC_SC_THEN_HIER_LEVEL | VARCHAR2 | Y | CLS | This is the product hierarchy level for THEN item for the mining for Anchor Subclass Top Affinities. The valid value is in ('SBC', 'CLS', 'DEPT'). |
| ANC_SC_MIN_SUPPORT_TXN_CNT | NUMBER | Y | 1000 | This optional configuration overrides the ARM_MIN_SUPPORT_TXN_CNT configuration and provides a lower boundary for the minimum supporting transaction count for the Anchor Subclass Top Affinities process. |

### Anchor Customer Segment Promotion Affinities (CUST_PROMO)

This mining service is identified as CUST_PROMO. The customer segment list that is used for mining is stored in the configuration table W_RTL_DMS_ATTR_LIST_G. The mining is executed against only one customer segment for each mining process, and only transactions that have that customer segment are used as source data. If any transaction has a promotion, the promotion event must in the range of the number of weeks that is defined in the parameter ARM_HIST_NUM_OF_WEEK. The "IF" column can have multiple values up to 3 subclasses. Organization hierarchy and rollup are not available from this mining output.

To set up the process queue:

```
begin
  rse_srvc_mgr.batch_pre_process('ARM', 'CUST_PROMO');
end;
```

**CUST_PROMO Configuration Parameters** The following parameters, listed in Table 7–5, are used for the CUST_PROMO service. They are configured in the table W_RTL_DMS_CONFIG_G, which is populated by an ETL process, and can be modified if necessary.

*Table 7–5 CUST_PROMO Service Configuration Parameters*

| Parameter | Type | Configurable | Sample Data | Description |
|---|---|---|---|---|
| CUST_PROMO_ATTR_LIST_CUST_SEG | VARCHAR2 | N | CUST_PROMO_ATTR_LIST_CUST_SEG_NUM | This is an identifier of the customer segment list to be processed from W_RTL_DMS_ATTR_LIST_G. |
| CUST_PROMO_CUST_SEG_RECLASS_IND | VARCHAR2 | Y | N | This indicates if any customer changed segment during the week. It is only manually updated by the end user. Setting this to 'Y' will cause mining program to re-execute the mining process against the whole history of mining data. |
| CUST_PROMO_IF_HIER_LEVEL | VARCHAR2 | N | SBC | This is the product hierarchy level for IF item for the mining for Anchor Customer Segment Promotion Affinities. |
| CUST_PROMO_MAX_DY_WID | NUMBER | N | 124020100214000 | This is the last date of data that Anchor Customer Segment Promotion Affinities mining program looks at. In most cases, it is the last date of the current week. |
| CUST_PROMO_MAX_SET_SIZE | NUMBER | Y | 3 | This is the total number of items in the Rule for Anchor Customer Segment Promotion Affinities. |
| CUST_PROMO_MIN_CONFIDENCE | NUMBER | Y | 0.05 | This is the minimum confidence filter for Anchor Customer Segment Promotion Affinities. |
| CUST_PROMO_MIN_DY_WID | NUMBER | N | 124020100214000 | This is the first date of data that Anchor Customer Segment Promotion Affinities mining program looks at. In most cases, it is the first date of current week. This will be reset to the first day of the week, that is, the number of weeks ago that is defined in the parameter ARM_HIST_NUM_OF_WEEK, when there is a reclassification of the product hierarchy in the current week. |
| CUST_PROMO_MIN_REVERSE_CONFIDENCE | NUMBER | Y | 0 | This is the minimum reverse confidence filter for Anchor Customer Segment Promotion Affinities. |
| CUST_PROMO_MIN_SUPPORT | NUMBER | Y | 0.0005 | This is the minimum support filter for Anchor Customer Segment Promotion Affinities. |
| CUST_PROMO_RESTART_HIST_IND | VARCHAR2 | Y | N | Valid value in ('Y', 'N'). |
| CUST_PROMO_THEN_HIER_LEVEL | VARCHAR2 | N | SBC | This is the product hierarchy level for THEN item for the mining for Anchor Customer Segment Promotion Affinities. |
| CUST_PROMO_MIN_SUPPORT_TXN_CNT | NUMBER | Y | 1000 | This optional configuration overrides the ARM_MIN_SUPPORT_TXN_CNT configuration and provides a lower boundary for the minimum supporting transaction count for the Anchor Customer Segment Promotion Affinities process. |

### Anchor Subclass Top Affinities by Promotions (ANC_SC_PROMO)

This mining service is identified as ANC_SC_PROMO. The promotion event list and organization list at one hierarchy level that is used for mining is stored in the configuration table W_RTL_DMS_ATTR_LIST_G. The mining is executed against only one organization for each mining process, and only transactions that have that organization are used as source data. The rollup of these results to higher organization hierarchy levels can be done at reporting time. The "IF" column can have multiple values up to three subclasses.

To set up the process queue:

```
begin
  rse_srvc_mgr.batch_pre_process('ARM', 'ANC_SC_PROMO');
end;
```

**ANC_SC_PROMO Configuration Parameters** The following parameters, listed in Table 7–6, are used for the ANC_SC_PROMO service. They are configured in the table W_RTL_DMS_CONFIG_G, which is populated by an ETL process and can be modified if necessary.

*Table 7–6 ANS_SC_PROMO Service Configuration Parameters*

| Parameter | Type | Configurable | Sample Data | Description |
|---|---|---|---|---|
| ANC_SC_PROMO_ATTR_LIST_ORG_DH | VARCHAR2 | N | ANC_SC_PROMO_ATTR_LIST_ORG_DH_NUM | This is an identifier of the organization list to be processed from W_RTL_DMS_ATTR_LIST_G. |
| ANC_SC_PROMO_ATTR_LIST_PROMO_COMP | VARCHAR2 | N | ANC_SC_PROMO_ATTR_LIST_PROMO_COMP_NUM | This is an identifier of the Promotion component list to be processed from W_RTL_DMS_ATTR_LIST_G. |
| ANC_SC_PROMO_IF_HIER_LEVEL | VARCHAR2 | N | SBC | This is the product hierarchy level for IF item for the mining for Anchor Subclass Top Affinities by Promotions. |
| ANC_SC_PROMO_MAX_DY_WID | NUMBER | N | 124020100214000 | This is the last date of data that Anchor Subclass Top Affinities by Promotions mining program looks at. In most cases, it is the last date of the current week. |
| ANC_SC_PROMO_MAX_SET_SIZE | NUMBER | Y | 3 | This is the total number of items in the Rule for Anchor Subclass Top Affinities by Promotions. |
| ANC_SC_PROMO_MIN_CONFIDENCE | NUMBER | Y | 0.05 | This is the minimum confidence filter for Anchor Subclass Top Affinities by Promotions. |
| ANC_SC_PROMO_MIN_DY_WID | NUMBER | N | 124020100214000 | This is the first date of data that Anchor Subclass Top Affinities by Promotions mining program looks at. In most cases, it is the first date of current week. This will be reset to the first day of the week, that is, the number of weeks ago which is defined in the parameter ARM_HIST_NUM_OF_WEEK, when there is a reclassification on product hierarchy in the current week. |

*Table 7–6 (Cont.) ANS_SC_PROMO Service Configuration Parameters*

| Parameter | Type | Configurable | Sample Data | Description |
|---|---|---|---|---|
| ANC_SC_PROMO_ MIN_REVERSE_ CONFIDENCE | NUMBER | Y | 0 | This is the minimum reverse confidence filter for Anchor Subclass Top Affinities by Promotions. |
| ANC_SC_PROMO_ MIN_SUPPORT | NUMBER | Y | 0.0005 | This is the minimum support filter for Anchor Subclass Top Affinities by Promotions. |
| ANC_SC_PROMO_ ORG_HIER_LEVEL | NUMBER | Y | REGION | This identifies the organization hierarchy level of this mining process. The valid values are in ('LOCATION', 'DISTRICT', 'AREA', 'CHAIN', 'REGION'). |
| ANC_SC_PROMO_ RESTART_HIST_IND | VARCHAR2 | N | N | Valid value in ('Y', 'N'). |
| ANC_SC_PROMO_ THEN_HIER_LEVEL | VARCHAR2 | N | SBC | This is the product hierarchy level for THEN item for the mining for Anchor Subclass Top Affinities by Promotions. |
| ANC_SC_PROMO_ MIN_SUPPORT_TXN_ CNT | NUMBER | Y | 1000 | This optional configuration overrides the ARM_MIN_SUPPORT_TXN_CNT configuration and provides a lower boundary for the minimum supporting transaction count for the Anchor Subclass Top Affinities by Promotions process. |

### Top 10 Promoted Subclass Affinities (TOP10_PROMO)

This mining service is identified as ANC_SC_PROMO. The mining process is executed against all transactions. The "IF" column can have multiple values up to 3 subclasses. Organization hierarchy and rollup are not available from this mining output.

To set up the process queue:

```
begin
  rse_srvc_mgr.batch_pre_process('ARM', 'TOP10_PROMO');
end;
```

**TOP10_PROMO Configuration Parameters** The following parameters, listed in Table 7–7, are used for the TOP10_PROMO service. They are configured in the table W_RTL_DMS_CONFIG_G, which is populated by an ETL process, and can be modified if necessary.

*Table 7–7 TOP10_PROMO Service Configuration Parameters*

| Parameter | Type | Configurable | Sample Data | Description |
|---|---|---|---|---|
| TOP10_PROMO_ ATTR_LIST_PROMO_ COMP | VARCHAR2 | N | TOP10_ PROMO_ ATTR_LIST_ PROMO_ COMP_NUM | This is an identifier of the Promotion component list to be processed from W_RTL_DMS_ATTR_LIST_G. |
| TOP10_PROMO_IF_ HIER_LEVEL | VARCHAR2 | N | SBC | This is the product hierarchy level for IF item for the mining for Current Top 10 Promoted Subclass Affinities. |
| TOP10_PROMO_MAX_ DY_WID | NUMBER | N | 124020100214 000 | This is the last date of data that Current Top 10 Promoted Subclass Affinities mining program looks at. In most cases, it is the last date of the current week. |

*Table 7–7 (Cont.) TOP10_PROMO Service Configuration Parameters*

| Parameter | Type | Configurable | Sample Data | Description |
|---|---|---|---|---|
| TOP10_PROMO_MAX_SET_SIZE | NUMBER | Y | 3 | This is the total number of items in the Rule for Current Top 10 Promoted Subclass Affinities. |
| TOP10_PROMO_MIN_CONFIDENCE | NUMBER | Y | 0.05 | This is the minimum confidence filter for Current Top 10 Promoted Subclass Affinities. |
| TOP10_PROMO_MIN_DY_WID | NUMBER | N | 124020100214000 | This is the first date of data that Current Top 10 Promoted Subclass Affinities mining program looks at. In most cases, it is the first date of current week. This is reset to the first day of the week that is, the number of weeks ago that is defined in the parameter ARM_HIST_NUM_OF_WEEK, when there is a reclassification on product hierarchy in the current week. |
| TOP10_PROMO_MIN_REVERSE_CONFIDENCE | NUMBER | Y | 0 | This is the minimum reverse confidence filter for Current Top 10 Promoted Subclass Affinities. |
| TOP10_PROMO_MIN_SUPPORT | NUMBER | Y | 0.0005 | This is the minimum support filter for Current Top 10 Promoted Subclass Affinities. |
| TOP10_PROMO_RESTART_HIST_IND | VARCHAR2 | N | N | Valid value in ('Y', 'N'). |
| TOP10_PROMO_THEN_HIER_LEVEL | VARCHAR2 | N | SBC | This is the product hierarchy level for THEN item for the mining for Current Top 10 Promoted Subclass Affinities. |

### Running ARM Services

The ARM Services have to be set up in the queue in the table RSE_SRVC_REQ_QUEUE the ARM Services can be run. The loading of the ARM configuration parameters and the execution of the various RSE_SRVC_MGR.BATCH_PRE_PROCESS steps as defined in the above sections should all be completed before processing the process queue, as shown below.

To run the process queue:

```
begin
  rse_srvc_mgr.process_queue('ARM');
end;
```

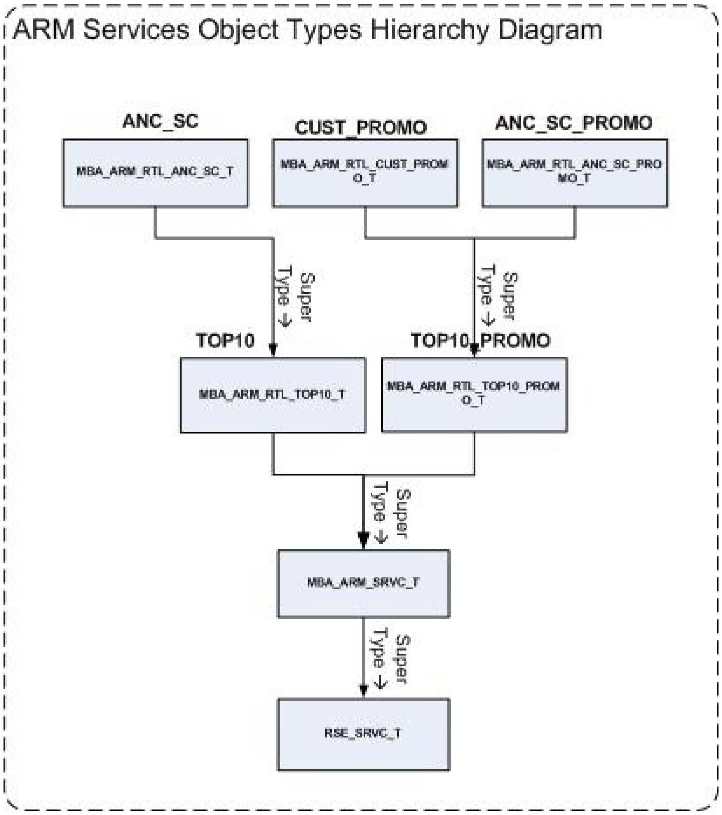
**ARM Services Objects Reference** Table 7–8 lists the input tables, output tables, and object types associated with ARM Services. Object types are user-defined extendable types, composed of two parts, attributes and methods, which enable encapsulation and code reuse during the various MBA operations. Refer to the *Oracle Database Application Developer's Guide*.

*Table 7–8 ARM Services Objects*

| Service Name | Object Type | Major Source Tables (input) | Staging Tables (output) |
|---|---|---|---|
| TOP10 | MBA_ARM_RTL_TOP10_T | W_RTL_SLS_TRX_IT_LC_DY_F | W_RTL_DMO_SLS_SC_WK_G, W_RTL_DMO_SLS_CL_WK_G, W_RTL_DMO_SLS_DP_WK_G |
| ANC_SC | MBA_ARM_RTL_ANC_SC_T | W_RTL_SLS_TRX_IT_LC_DY_F | W_RTL_DMO_SLS_SC_WK_SUP_G |
| CUST_PROMO | MBA_ARM_RTL_ANC_CUST_PROMO_T | W_RTL_SLS_TRX_IT_LC_DY_F | W_RTL_DMO_SLSPR_CUST_SC_WK_G |
| ANC_SC_PROMO | MBA_ARM_RTL_ANC_SC_PROMO_T | W_RTL_SLS_TRX_IT_LC_DY_F | W_RTL_DMO_SLSPR_SC_WK_SUP_G |
| TOP10_PROMO | MBA_ARM_RTL_TOP10_PROMO_T | W_RTL_SLS_TRX_IT_LC_DY_F | W_RTL_DMO_SLSPR_SC_WK_G |

**ARM Services Objects Types Hierarchy** Figure 7–4 shows the hierarchy of ARM Services object types.

*Figure 7–4 Market Basket Analysis ARM Services Object Types Hierarchy*



## Attribute List Table: w_rtl_dms_attr_list_g

Table 7–9 contains the name and value of each attribute processed for the data mining process. The names of the attribute lists are defined in the table W_RTL_DMS_CONFIG_G.

*Table 7–9 Data Mining Process Attributes*

| Attribute Description | Attribute Name (from W_RTL_DMS_CONFIG_G) | Attribute Value |
|---|---|---|
| Anchor Subclass: List of subclasses that are used for supervised non-promotion related product affinity. | Value defined for ANC_SC_ATTR_LIST_SBC parameter. | Mining program needs to join the W_RTL_DMS_PRODUCT_G to get attribute value. |
| Promotional Component: The list of promotions that are used for promotion-related supervised mining. | Value defined for TOP10_PROMO_ATTR_LIST_PROMO_COMP parameter. | Mining program needs to join the promotion table W_RTL_PROM_D to get attribute value. |
| Customer Segment: The list of customer segments. | Value defined for CUST_PROMO_ATTR_LIST_CUST_SEG parameter. | Mining program needs to join customer table to get relationship between customer and customer segment RSE_CUR_CUSTSEG_D. |
| Organization: The list of organization hierarchy values that are used for mining. The hierarchy level is defined in W_RTL_DMS_CONFIG_G table where PARA_NAME = 'MINING_ORG_LEVEL_PROMO_SUP. | Value defined for ANC_SC_PROMO_ATTR_LIST_ORG_DH parameter. | Mining program needs to join the W_RTL_DMS_ORG_DH_G and W_RTL_DMS_INT_ORG_DH_G to get value. |
| Anchor Subclass Promotion component: List of subclasses that are used for supervised promotion-related product affinity. | Value defined for ANC_SC_PROMO_ATTR_LIST_PROMO_COMP parameter. | Mining program needs to join the promotion table W_RTL_PROM_D to get value. |

Table 7–10 lists the sample data for the attributes:

*Table 7–10 Attribute Sample Data*

| Attribute Name | Attribute Value |
|---|---|
| ANC_SC_ATTR_LIST_SBC_NUM | 57~26~5425 |
| TOP10_PROMO_ATTR_LIST_PROMO_COMP_NUM | 652663 |
| CUST_PROMO_ATTR_LIST_CUST_SEG_NUM | FreqHigh |
| ANC_SC_PROMO_ATTR_LIST_ORG_DH_NUM | A4 |
| ANC_SC_PROMO_ATTR_LIST_PROMO_COMP_NUM | 652808 |

## Baseline Services

The Baseline services process sales transaction from Retail Analytics into a suitable structure and calculates non-promoted baseline sales for items that are promoted.

Baseline is a process of calculating non-promoted baseline sales for items that are promoted. It transfers sales data by week from Retail Analytics tables, identifying the appropriate weeks that are suitable to be included in the baseline calculation.

For instance, item/location weekly sales are suitable for baseline calculation only if they does not have promotion sales for the week. In the case of overlapping promotions in a given week due to weekly sales by promotion transfer from Retail Analytics, baseline values split proportionally across the overlapping promotions using sales results from each promotion. Overlapping in promotions whenever the same item/location appears on more than one promotion for the same day within the week are also considered. In case of promotions that could run on the same week but different days, the baseline is adjusted only based on the number of days each promotion ran within the week.

Once the data is ready for calculation, the process performs the baseline calculation for all the promotions that have completed. Baseline results by promotion are saved to the baseline staging table W_RTL_DMO_SLS_BL_PC_IT_LC_WK_G. Upon completion of baseline processing, control is returned to Retail Analytics in order to perform data integration to the final target table W_RTL_SLSPR_BL_PC_IT_LC_WK_A.

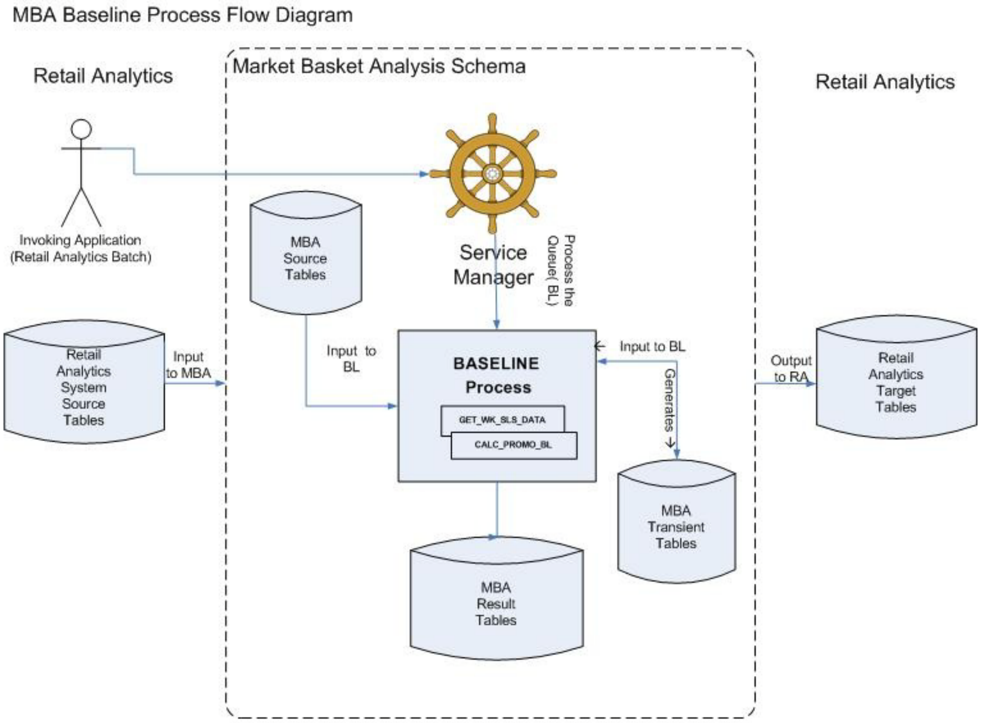
There are two services associated with Baseline.

- Gathering weekly sales data (GET_WK_SLS_DATA)
- Calculating Baseline (CALC_PROMO_BL)

Since the Calculating Baseline (CALC_PROMO_BL Service) depends on the data that are gathered (GET_WK_SLS_DATA Service), the services should be executed in the specified order shown in the list.

Figure 7–5 shows the MBA Baseline Process flow diagram.

***Figure 7–5 Market Basket Analysis MBA Baseline Process Flow***



## Running Promotion Baseline Service

Here is the method for the data gathering process of relevant sales data into MBA. This process must be completed successfully before you execute the CALC_PROMO_BL step.

To set up the process queue for gathering Baseline Weekly Sales Data:

```
begin
  rse_srvc_mgr.batch_pre_process('BL', 'GET_WK_SLS_DATA');
end;
```

To process the service queue:

```
begin
  rse_srvc_mgr.process_queue('BL');
end;
```

Upon successful completion of this process, the following can be run to calculate the Promotion Baseline data and publish it to the staging table.

To set up the process queue for Baseline calculation:

```
begin
  rse_srvc_mgr.batch_pre_process('BL', 'CALC_PROMO_BL');
end;
```

To process the queue that was set for Baseline Calculation:

```
begin
  rse_srvc_mgr.process_queue('BL');
end;
```

### Baseline Configuration

Table 7–11 lists the configuration values that must be defined within the W_RTL_DMS_CONFIG_G table for the baseline processing. They are configured during initial configuration and are loaded by the ETL, and can be modified if necessary.

***Table 7–11 Baseline Configuration Values***

| Parameter | Type | Required | Sample Data | Description |
|---|---|---|---|---|
| BL_WEEKS_TO_USE | NUMBER | Y | 16 | Number of weeks to use for baseline calculation. Number of sales data weeks to use before the promotion start is determined as CEIL (BL_WEEKS_TO_USE/2) and number of sales data weeks to use after the promotion ends is determined as TRUNC (BL_WEEKS_TO_USE/2). |
| BL_POST_PROMO_CALC_FREQ | NUMBER | Y | 2 | Baseline calculation always occur the week(s) the promotion is active, the week after the promotion completes, and at the end of the after promotion weeks. This parameter indicates the recalculation frequency for the weeks in the middle (end of promotion to end of post promotion weeks). That is, a value of 1 indicates a recalculation every week, a value of 2 indicates a recalculation every 2 weeks, and so on. |
| BL_WEEK_TO_PROCESS | NUMBER | Y | 124020100003 | Indicates "the" baseline processing week used to calculate the baseline using sales data around this week. |
| BL_WEIGHT_AGE_BASE | NUMBER | Y | 0.5 | This configuration value is used within the baseline calculation on this expression, where "a" corresponds to BL_WEIGHT_AGE_BASE, with a default value of 0.5, and the exponent "n" corresponds to the number of weeks between the promotional week and the adjacent weeks. The adjacent weeks are capped to the PRE and POST promotion number. |

*Table 7–11 (Cont.) Baseline Configuration Values*

| Parameter | Type | Required | Sample Data | Description |
|---|---|---|---|---|
| BL_WK_SALES_ DATA_DOP | NUMBER | N | 4 | Number of recommended parallel threads to use for weekly baseline calculation. Weekly baseline calculation retrieves sales data from Retail Analytics for multiple weeks. This step can be parallelized by the use of this parameter. |
| BL_PROMO_CALC_ DOP | NUMBER | N | 4 | Number of concurrent threads to use while doing promotion baseline calculation. This service can be parallelized by the use of this parameter. |
| BL_PRE_PROMO_ WEEKS | NUMBER | N | 5 | This parameter can be used to specifically indicate the number of sales weeks prior to the promotions that should be used for baseline calculation. This parameter takes precedence over any other value used to identify the number of PRE/POST promo weeks. |
| BL_POST_PROMO_ WEEKS | NUMBER | N | 9 | This parameter can be used to specifically indicate the number of sales weeks after the promotions that should be used for baseline calculation. This parameter takes precedence over any other value used to identify the number of PRE/POST promo weeks. |
| BL_REFRESH_PRE_ PWEEKS | NUMBER | N | 2 | Baseline calculation extracts sales data from Retail Analytics and preserves the aggregated data within MBA for future use. The data can miss back posted and return transactions that might be relevant for future calculations. This parameter indicates the number of the most recent weeks prior to the processing week that needs to be refreshed. (For example, a value of 2 indicates the baseline process that the two most recent weeks need to be refreshed from Retail Analytics, a value of 0 indicates that historical data already available on MBA schema must be used, effectively ignoring returns and back posted data, and so on). |

## Parallelism

MBA provides two levels of parallelism in accessing the services with the help of the table RSE_SRVC_REQ_QUEUE table, the object type RSE_SRVC_T, and the RSE_ SRVC_MGR package with the help of dbms_parallel_execute. The first level allows the processing of different Service Request Groups in parallel threads and the second level allows parallel processing of the different service requests within a given Service Request Group.

Parallel processing depends on how the parameters associated with parallel processing are configured in the W_RTL_DMS_ATTR_LIST_G table. For ARM Services, it depends on the degree of ARM data mining batches that can be simultaneously processed within each week (ARM_BATCH_DOP) and degree of weeks that can be simultaneously processed for ARM data mining (ARM_WEEKLY_ DOP), whereas for Baseline, it depends on the number of parallel threads for sales data retrieval (BL_WK_SALES_DATA_DOP), for GET_WK_SLS_DATA service, and the number of parallel threads for calculating Baseline (BL_PROMO_CALC_DOP) for CALC_PROMO_BL (Refer Baseline Configuration).

The maximum number of parallel threads that can be used depends on the particular server and database configurations. Ultimately, the limit on the number of parallel threads that can be used is controlled by the database configuration for the maximum number of jobs allowed (job_queue_processes).

Enabling parallel processing of ARM at both levels should only be done in cases where there is a sufficiently large database server with sufficient IO capabilities. For example, if you specify two parallel weeks (ARM_BATCH_DOP =2)and three parallel batches (ARM_WEEKLY_DOP=3), then you need the support to run eight concurrent jobs (2 weeks + (2 weeks * 3 batches per week)) in the database. In addition, if parallel processing for Baseline is enabled, then support for that many threads of processing is also required. As Baseline service are serial (unlike ARM), the configuration of four (BL_WK_SALES_DATA_DOP=4 and BL_PROMO_CALC_DOP=4) would require a maximum of four concurrent jobs, and it would not compound like it does for ARM. In this specific case, if Baseline runs simultaneously with ARM, the database server needs to accommodate both needs, that is, for 12 (8 ARM + 4 Baseline) concurrent jobs.

## Maintenance

This section contains information about MBA maintenance.

### Purging ODM Models: purge_odm_models

This routine purges the ODM model and its supporting DB tables, which helps to remove any models that may have been left over from a failed process that was never fully recovered. To remove all leftover models related to the MBA processing, purge_odm_models can be invoked as shown below to remove all models that follow the specified naming pattern. The list of models eligible for purging can be obtained by querying the data dictionary view USER_MINING_MODELS in the MBA schema.

***Table 7–12 Purging ODM Models***

| Input Parameter | Type | Description |
|---|---|---|
| p_odm_model_name | VARCHAR2 | Name corresponding to the model to be removed. Optional parameter. |

Example: To purge a ODM model named like 'TMP$ARM_BATCH'

```
begin
  rse_srvc_maint.purge_odm_models('TMP$ARM_BATCH%');
end;
```

## Market Basket Analysis Terms

This section defines some relevant terms in Table 7–13.

***Table 7–13 MBA Terminology***

| Term | Definition |
|---|---|
| ARM | Association Rule Mining |
| DM | Data Mining |
| ODM | Oracle Data Mining algorithm |
| Antecedent | A set of item(s) that are frequently associated with another item. This is also known as the IF part of a Rule, or the Left Hand Side (LHS) of a rule. |
| Consequent | An item that has been found to be associated with another set of item(s). This is also known as the THEN part of a Rule, or the Right Hand Side (RHS) of a rule. |

*Table 7–13 (Cont.) MBA Terminology*

| Term | Definition |
|---|---|
| Support | The number of occurrences that the corresponding item(s) were found in a transaction. |
| Confidence | Expresses how likely the Consequent will be found in transactions which contain the Antecedent. |
| Lift | How many times more often the Antecedent and Consequent occur together then they would have if they were statistically independent. |
| BL | Baseline |
| ETL | Extraction, Transformation and Loading |
| DOP | Degree of Parallelism |

# 8

# Server Configuration

This chapter addresses the configuration of the Coherence Server. It contains the following section:

- ORASE Coherence Server Configuration

## ORASE Coherence Server Configuration

In ORASE AC and ASO, if one computational node is insufficient to withstand the required computational load, Oracle Coherence cluster can be used to scale horizontally. Oracle Coherence is an Oracle product that provides a distributed grid and cache solution. The ORASE Coherence Server can be used by the related ORASE calculation modules.

### ORASE Coherence Server Startup Script

The ORASE Coherence Server is started by the following shell script:

```
start_rse_coherence_srv.ksh
```

which is installed here:

```
<RSE_HOME>/common/RseCoherenceServer/scripts
```

This shell script can be adjusted to specify Java runtime parameters, such as

```
-Dtangosol.coherence.clusteraddress=<your_value>
```

The started process has to be monitored and restarted as needed.

All ORASE-specific xml configuration can be done in the corresponding xml files, which are installed here:

```
 <RSE_HOME>/common/RseCoherenceServer/config/common
```

The following files are configurable:

- log4j.xml
- tangosol-coherence-override.xml
- jp-server-app-context.xml
- META-INF/persistence.xml

This file does not need to be modified.

- jpdb-application-context.xml

After making any necessary configuration changes, you can start the ORASE Coherence Server by following these steps:

1. Change directory to <RSE_HOME>/common/RseCoherenceServer/scripts

   ```
   cd <RSE_HOME>/common/RseCoherenceServer/scripts
   ```

2. Execute start_rse_coherence_srv.ksh

   ```
   nohup ./ start_rse_coherence_srv.ksh &
   ```

3. Review the following two logs as necessary:

   ```
   <RSE_HOME>/common/RseCoherenceServer/scripts/bin/rse_server_stdout.log
   <RSE_HOME>/common/RseCoherenceServer/scripts/bin/rse_coherence_server.log
   ```

## Logging Configuration

The log configuration file is located here:

```
<RSE_HOME>/common/RseCoherenceServer/config/common/log4j.xml
```

See http://logging.apache.org/log4j/2.x/manual/configuration.html#XML for details on configuration options.

## Coherence Server Configuration

The Coherence Server configuration file is located here:

```
<RSE_HOME>/common/RseCoherenceServer/config/common/tangosol-coherence-override.xml
```

The default configuration for the ORASE Coherence Server is a Multicast Listener.

```
<multicast-listener>
        <address>224.10.0.0</address>
        <time-to-live>1</time-to-live>
        <join-timeout-milliseconds>1000</join-timeout-milliseconds>
</multicast-listener>
```

If multiple instances of the RSE Coherence Server are being run, the Multicast Listener Address must be unique for each instance of RSE Coherence Server and the respective ORASE Applications. The file that must be edited (and in sync between RSE Coherence Server and ORASE Applications) is tangosol-coherence-override.xml.

This file is located here:

```
<RSE_HOME>/common/RseCoherenceServer/config/common/tangosol-coherence-override.xml
```

For ORASE AC, the file is located here:

```
<RSE_
HOME>/cdm/cis/application/CisSharedLib.war!WEB-INF/classes/tangosol-coherence-over
ride.xml
```

For ASO, the file is located here:

```
<RSE_HOME>/so/application/SpaceOptimization_
application1.ear!soviewcontroller.war!WEB-INF/classes/tangosol-coherence-override.
xml
```

After editing this file (keeping the <address> in sync), you must update the CisSharedLib.war and SpaceOptimization_application1.ear in the WLS Domain.

**Note:** If you cannot edit the deployed ear/wars in place, then you can expand CisSharedLib.war and SpaceOptimization_application1.ear!soviewcontroller.war manually and modify the files. CisSharedLib.war and SpaceOptimization_application1.ear!soviewcontroller.war must be then reconstituted properly.

See http://docs.oracle.com/cd/E18686_01/coh.37/e18677/appendix_operational.htm#BABHDDEH for details on configuration options.

## ORASE Job Processor Configuration

The ORASE Job Processor configuration file is located here:

```
<RSE_HOME>/common/RseCoherenceServer/config/common/jp-server-app-context.xml
```

Modify this file to specify the number of threads to run:

```
<constructor-arg value="4"/> <!-- number of threads -->
```

The default value is 4.

## Configure Database Connection for Coherence Server

<RSE_HOME>/common/RseCoherenceServer/config/common/META-INF/persistence.xml contains the Database Configuration details. The Oracle Wallet alias is used to connect the Coherence Server to the Database. This file is modified by the installer during installation, as %{RSE_DB_BATCH_USER}% is replaced with the appropriate value.

```
<property name="javax.persistence.jdbc.url"
value="jdbc:oracle:oci:/@%{RSE_DB_BATCH_USER}%"/>
```

## Update RSE_PROC_TASK_TMPL to Define Which Execution Mode to Use

When using the ORASE Coherence Server, update the table RSE_PROC_TASK_TMPL.

For ASO, execute this SQL:

```
update RSE_PROC_TASK_TMPL set PKG='coherenceAsynchService' where id = 2001001;
commit;
```

For ORASE AC, execute this SQL:

```
update RSE_PROC_TASK_TMPL set PKG='coherenceAsynchService' where id = 211;
commit;
```

After updating RSE_PROC_TASK_TMPL, you must restart the WLS Domain.

## Server Adapters

The following service adapters are available:

*Table 8–1 Common Service Adapters*

| Bean ID | Description |
|---|---|
| coherenceAsynchService | Asynchronously process in the Coherence grid |

*Table 8–1 (Cont.) Common Service Adapters*

| Bean ID | Description |
|---|---|
| localConcurrentService | Asynchronously process locally |
| localSynchServiceNoDbLog | Synchronously process a step |

*Table 8–2 ASO Adapters*

| Bean ID | Description |
|---|---|
| soLocalConcurrentService | Asynchronously process optimization requests |

*Table 8–3 DT Adapters*

| Bean ID | Description |
|---|---|
| dtLocalConcurrentService | Asynchronously process DT stages |
| dtLocalSynchConcurrentService | Synchronously process DT model apply |
| dtBatchLocalConcurrentService | Asynchronously process DT export |

*Table 8–4 CDT Adapters*

| Bean ID | Description |
|---|---|
| cdtLocalConcurrentService | Asynchronously process CDT stages |
| cdtLocalSynchConcurrentServiceNoDbLog | Synchronously process a CDT step |

*Table 8–5 Adding New Adapters*

| Field | Description |
|---|---|
| ID | A unique bean name |
| Class | Class: oracle.retail.rse.common.jobprocessor.service.LocalServiceAdapter - Local (a)sync |
| Constructor-arg | logInDb : true or false<br>useCurrentThread : true or false<br>useCurrentThreadfixedThreadPoolSize : number of threads in a fixed pool or 0 to allow automatic grow / reduce |

```
<bean id="serviceName"
    class="oracle.retail.rse.common.jobprocessor.service.LocalServiceAdapter"
    scope="singleton" lazy-init="true" destroy-method="close">
        <constructor-arg name="logInDb" value="false"/>
        <constructor-arg name="useCurrentThread" value="true"/>
        <constructor-arg name="fixedThreadPoolSize" value="2"/>
</bean>
```

As needed, new adapters can be added by entering the necessary information into the application's configuration XML. The application specific configuration files are located in directories within the <RSE_HOME>common/RseCoherenceServer/config directory. For example:

Here is an example of a service that does not create a record in the database, works synchronously, and uses two threads in a fixed thread pool:

Using stage-based execution. In the database, insert a root record into rse_proc_tmpl. Insert matching records into rse_proc_task_tmpl. Examples can be found in db/seed_data/rse_proc_task_tmpl.ctl files, this set is from DT:

***Table 8–6 Example***

| Name | Description |
|---|---|
| ID | It must be unique. A typical convention is to concatenate rse_proc_tmpl with 0 ; 1,... to show the tree-based dependency between the root entry, stages, and steps. |
| PROC_ID | It must match rse_proc_tmpl.id. |
| NAME | Free-form. It typically has a name of the process and stage or step number. |
| DESCR | Free-form. |
| STEP_NUM | It must be null for the first entry (a zero step). |
| PARENT_ID | The ID of the parent rse_proc_task_tmpl entry. It is null for the first entry (a zero step). |
| EXEC_PROC_ID | Not used / reserved. |
| SRVC_TYPE_ID | Not used / reserved. |
| SRVC_NAME | Not used / reserved. |
| PKG | A key word stage for a stage or the spring bean name for the adapter that runs this step. |
| PROC | N/A for a stage or the spring bean name that runs as part of this step. |
| DELETE_FLG | Null if not deleted. |
| ACTIVE_FLG | A value of T if this record is active. |
| MAX_RUN_TIME_MIN | Not used / reserved. |
| MAX_FAILURES | Not used / reserved. |
| RESTARTABLE_FLG | Not used / reserved. |

Example:

```
"810"|"81"|"DT-RDF-Start"|"DT RDF - starting
step"|""|""|""|""|""|"dtBatchLocalConcurrentService"|"dtRdfExportJob"|""|"T"|""|""
|""
"8101"|"81"|"DT-RDF-Stage1"|"DT RDF
-Stage1"|"1"|"810"|""|""|""|"stage"|"n/a"|""|"T"|""|""|""
"810110"|"81"|"DT-RDF-Stage1-step1"|"DT RDF
-Stage1-step1"|"1"|"8101"|""|""|""|"localSynchServiceNoDbLog"|"dtPlsqlJob"|""|"T"|
""|""|""
"8102"|"81"|"DT-RDF-Stage2"|"DT-RDF-Stage2"|"2"|"810"|""|""|""|"stage"|"n/a"|""|"T
"|""|""|""
"810210"|"81"|"DT-RDF-Stage2-step1"|"DT-RDF-Stage2-step1"|"1"|"8102"|""|""|""|"dtL
ocalSynchConcurrentServiceNoDbLog"|"dtRdfJob"|""|"T"|""|""|""
"81021010"|"81"|"DT-Stage2-step1-substep1"|"DT-RDF-Stage2-step1-substep1"|"1"|"810
210"|""|""|""|"localSynchServiceNoDbLog"|"dtRdfCalcJob"|""|"T"|""|""|""
"8103"|"81"|"DT-RDF-Stage3"|"DT-RDF-Stage3"|"3"|"810"|""|""|""|"stage"|"n/a"|""|"T
"|""|""|""
"810310"|"81"|"DT-RDF-Stage3-step1"|"DT-RDF-Stage3-step1"|"1"|"8103"|""|""|""|"loc
alSynchServiceNoDbLog"|"dtPlsqlJob"|""|"T"|""|""|""
```

A corresponding <RSE_HOME>/common/RseCoherenceServer/config/cdm/dt/dttojp-application-context.xml has the spring beans defined.

Predefined wrapper jobs: common jobs have been defined. The following is for a step wrapper job:

```
 <bean id="dtRdfJob"
class="oracle.retail.rse.common.jobprocessor.job.StepWrapperJob"
    scope="prototype">
         <constructor-arg> <!-- properties -->
            <map>
                <entry key="CHECK_CANCELLED"><value>false</value></entry>
            </map>
         </constructor-arg>
</bean>
```

For a call that accepts a JDBC connection and an ID pointer of type long (to be executed as a step. Database entry examples are above):

```
<bean id="dtRdfCalcJob"
class="oracle.retail.rse.common.jobprocessor.job.StepCalcWrapperJob"
    scope="prototype">
         <constructor-arg> <!-- properties -->
            <map>
                <entry key="CALL_
CLASS"><value>oracle.retail.rse.cdm.dt.rdf.ModelApplyForRDF</value></entry>
                <!-- the signature must be Connection, long -->
                <entry key="CALL_
METHOD"><value>skuStoreMultipliersforRDF</value></entry>
            </map>
         </constructor-arg>
</bean>
```

There are also PL/SQL execution beans, For example (database entry examples are above):

```
<bean id="dtPlsqlJob" class="oracle.retail.rse.common.jobprocessor.PlsqlJob"
    scope="prototype">
    <constructor-arg ref="dtPlsqlJobConfig"/>
</bean>
<util:map id="dtPlsqlJobConfig">
    <entry key="510110"><bean
class="oracle.retail.rse.common.jobprocessor.PlsqlConfig">
                <constructor-arg >
                <value><![CDATA[declare v_id NUMBER;
                begin v_id := ?;
                end;]]></value></constructor-arg>
                <constructor-arg>
                    <list>
                        <value>batchId</value>
                    </list>
                </constructor-arg>
                </bean>
            </entry> …
```

Running through a batch script. Define the parameters in *-application-context.xml. For example:

```
<!-- param name - description -->
<bean id="dtRdfBatchParams" class="java.util.HashMap"
    scope="singleton">
    <constructor-arg>
        <map key-type="java.lang.String" value-type="java.lang.String">
            <entry key="userId" value="User who runs this batch"/>
            <entry key="versionId" value="Version ID to export (optional)"/>…
```

Provide a wrapper shell script. For example, (from dt_rdf_export.ksh):

```
P="-beanName dtRdfBatchParams -userId ${USER} -appName DT -procName DT_RDF
-batchRunnerName RdfExportRunner -waitTime 5 -timeout 900"
Q="$JAVA_HOME/bin/java -cp $CP
oracle.retail.rse.common.jobprocessor.job.RunnableInStagesServiceHelper ${P}"
```

Implementation-reserved JobType enum entries. Insert matching records into the database and use in the code as appropriate:

```
IMPL_0("IMPL", "IMPL_0"),
IMPL_1("IMPL", "IMPL_1"),
IMPL_2("IMPL", "IMPL_2"),
IMPL_3("IMPL", "IMPL_3"),
IMPL_4("IMPL", "IMPL_4"),
IMPL_5("IMPL", "IMPL_5"),
IMPL_6("IMPL", "IMPL_6"),
IMPL_7("IMPL", "IMPL_7"),
IMPL_8("IMPL", "IMPL_8"),
IMPL_9("IMPL", "IMPL_9");
```

The following tables are used:

- PROCESS_STATUS. Reference table for the status IDs and descriptions.
- RSE_PROC_TMPL. Batch description (metadata): the overall list of tasks.
- RSE_PROC_TASK_TMPL. Job description (metadata): task steps.

  Core fields:

  - ID, PROC_ID. To be referred to from the Java / middle-tier code.
  - PKG. Spring bean name of the service adapter that runs the job.
  - PROC. Spring bean name of the actual job Java bean to execute.
- E_PROC_REQ_QUEUE. Batch runtime (note that final status is only updated for a staged-type executions).
- RSE_PROC_REQ_TASK_QUEUE. Job runtime, status.

# 9

# 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 RA and are derived from product descriptions and merchandise hierarchy.

RADM may or may not contain product attributes and any attributes found in RADM may have been created for BI reporting or other purposes and may need mining or preprocessing to make them suitable for ORASE.

Some steps in attribute preprocessing require manipulating attribute data. Oracle Enterprise Product Data Quality is a 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: Dannon non-fat organix 6 oz.
- Class description: Dairy product.
- Sub-class description: Yogurt.

SKU/Store attributes determined by preprocessing:

- Brand
- Price
- Size

Note that CDT and DT modeling work 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 and DT, it is better to define between 3 and 5 prices bins (that is, budget, regular, premium, and elite).

### Process Overview

The basic steps for attribute preprocessing are as follows:

- Populating RADM with attribute data
- Translating (optional)

- Parsing
- Cleansing and standardizing
- Categorizing and labeling
- Defining attributes
- Binning and grouping

### Populating RADM with Attribute Data

To make RADM attributes suitable for ORASE requires a few steps for the applications to use this data.

The first requirement is to ensure that the attribute values are populated in RADM. This is the source for ORASE's attribute data and must be loaded there in order to be available to ORASE.

Regarding RADM attributes: In RADM, an attribute can be defined in multiple ways. Flex attributes are typically stored in a column of the W_PRODUCT_ATTR_D table. RADM 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.

RA also offers the ability to store Item Differentiators for products. These are essentially User Defined Attributes (UDAs), which consist of lookup code for the attribute and the attribute value. These lookup codes are then defined in RADM's 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_GRP 1_D table.

Once attributes are available in RADM, it is necessary to define these attributes in ORASE's RSE_BUSINESS_OBJECT_ATTR_MD table. This table must be set up with appropriate metadata to define the source of the attributes from RADM. The sample seed_data file for this table contains some standard attributes that would be defined in RADM, but the table needs to be adjusted to contain the complete list of attributes that should be available for ORASE modules to use. This needs to include Flex Attributes as well as User Defined Attributes.

Once attributes are defined in ORASE's 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 would be in RSE_BUSINESS_OBJECT_ATTR_MD). The second table of the interface (RSE_PROD_ATTR_VALUE_XREF_STG), would enable 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 "Dannon", "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 "Dannon" 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 are the categories.

### Binning and Grouping

Binning and grouping is used to consolidate and reduce the number of possible values for an attribute into a manageable number.

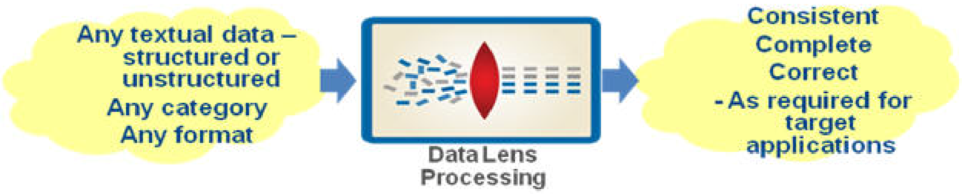
- Binning divides numerical attributes, such as 'price', 'discounts', and 'mileage.' into discrete sets of ranges, such as, '<=$10', '$10~$25', and '>$25'.
- Grouping combines textual attributes that are too granular into a smaller set of attribute values. For example, 'tea weight' can have dozens of values; grouping merges the values into coarser ranges (like "small" or "large") and reduces the number of possible attribute values.

## Enterprise Data Quality for Product Data (EDQP)

Enterprise Data Quality for Product is a pre-built solution that processes textual data through semantic recognition and cleanses, parses, and classifies data into well-formatted standardized text for functions like attribute extraction. It is made up of three core modules that work together to enforce category-specific standards on disparate product information:

- Oracle DataLens Knowledge Studio
- Oracle DataLens Application Studio
- Oracle DataLens Governance Studio

*Figure 9–1 DataLens Preprocessing*


Any textual data – structured or unstructured
Any category
Any format
Data Lens Processing
Consistent
Complete
Correct
- As required for target applications

- Semantic model recognizes item category based on context.
- Target information is identified and extracted.
- Missing information is flagged.
- Items are transformed and reassembled to meet target system standards.

For attribute preprocessing, the main application of EDQP is the Knowledge Studio. In Knowledge Studio, input data can be any textual data, structured or unstructured in various categories or formats. For more information on EDPQ, see *Oracle Enterprise Data Quality for Product Data* (E24155-02).

## Product Attribute Loading

This section provides an example of adding an attribute for use by ORASE 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 RADM
3. Add attribute definition in ORASE tables, if it not already present
4. Run the process to load attribute data from RADM
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. Process the interface staging tables
9. Update CIS attribute data to reflect the new attribute (product attributes)
10. Update 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 ORASE.

### Determine Attribute Source and Define in ORASE tables

The new attribute is loaded from RADM for each of the products that require this attribute. RADM has multiple ways of loading attributes, so the approach used varies, depending on where and how the data is stored in RADM. 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

RADM'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 for the ORASE, these tables are effectively the same and require the same type of handling.

These tables each have similar entries in the RSE_BUSINESS_OBJECT_DB_SRC, and no change should be required here. However, a new row is required in RSE_BUSINESS_OBJECT_ATTR_MD that refers to the proper data source table. For rows added to RSE_BUSINESS_OBJECT_ATTR_MD, the values must be populated as follows:

*Table 9–1 RSE_BUSINESS_OBJECT_ATTR_MD*

| Column | Example | Description |
|---|---|---|
| ID | 100 | Unique ID for this entry. |
| BUSINESS_OBJECT_MD_ID | 1 | Foreign Key to RSE_BUSINESS_OBJECT_MD. For product attributes, this should be a 1. |
| BUSINESS_OBJECT_DB_SRC_ID | 8 | Foreign Key to RSE_BUSINESS_OBJECT_DB_SRC which relates to the table that contains the new attribute column. |
| NAME | Flavor | A short descriptive name for the attribute. |
| DESCR | Flavor | A more descriptive name for the attribute. |
| SHORT_DB_NAME | Flavor | An alias for this database column name, which can be used as the column name for this attribute value when the attribute is later used in a cross tab query. |
| SRC_COLUMN_NAME | PRODUCT_ATTR25_NAME | The name of the column that contains this attribute. |
| EXCLUDE_FLG | N | A Y/N value to either exclude this attribute (Y) or not (N) from processing. |

### W_RTL_ITEM_GRP1_D or W_RTL_ITEM_GRP2_D

The W_RTL_ITEM_GRP1_D and W_RTL_ITEM_GRP2_D tables in RADM are different than the other product attribute sources, in that these tables can have attributes implemented as unique rows and specific columns. These tables contain 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 WID columns (ex. BRAND_WID) are also possible.

For processing new attributes in this table, it may be necessary to add new rows to the RSE_BUSINESS_OBJECT_DB_SRC table if any special rules regarding WHERE clauses for filtering data for the attribute are required. If an attribute is defined with a PROD_

GRP_TYPE of ITEMUDA, BRAND, COLOR, FLAVOR, SCENT, or SIZE, then no additional rows will be required, as these should already be present. If a new row is required because there is no predefined row with a suitable FILTER_CLAUSE, then add a row as necessary, adjusting the FILTER_CLAUSE appropriately, using one of the existing rows related to W_RTL_ITEM_GRP1_D as an example.

The next step is to define the attribute in the RSE_BUSINESS_OBJECT_ATTR_MD table. Examples for setting up data are:

***Table 9–2 RSE_BUSINESS_OBJECT_ATTR_MD***

| Column | Example | Description |
|---|---|---|
| ID | 101 | Unique ID for this entry. |
| BUSINESS_OBJECT_MD_ID | 1 | Foreign Key to RSE_BUSINESS_OBJECT_MD. For product attributes, this should be a 1. |
| BUSINESS_OBJECT_DB_SRC_ID | 10 | Foreign Key to RSE_BUSINESS_OBJECT_DB_SRC that relates to the table that contains the new attribute column. |
| NAME | Flavor | A short descriptive name for the attribute. |
| DESCR | Flavor | A more descriptive name for the attribute. |
| SHORT_DB_NAME | FLAVOR | An alias for this database column name, which can be used as the column name for this attribute value when the attribute is later used in a cross tab query. |
| SRC_COLUMN_NAME | FLEX_ATTRIB_3_CHAR | The name of the column that contains this attribute. For ITEMUDA, the actual attribute value is expected to be found in FLEX_ATTRIB_3_CHAR. Adjust if needed. |
| PROD_GRP_TYPE | ITEMUDA | This column contains the value as found in the same column of W_RTL_ITEM_GRP1_D in RADM. |
| ATTR_EXT_CODE | FLAVOR | This value is normally the same value as the one in FLEX_ATTRIB_1_CHAR and is used to define the external code for the attribute. |
| TL_DOMAIN_CODE | ITEM_UDA | For ITEMUDA's, the attributes are normally lookup codes, which therefore can have translated values associated with the codes. This value defines how to locate such values from the W_DOMAIN_MEMBER_LKP_TL table in RADM. |
| TL_JOIN_EXPR | pa.attr_value_ext_code\|\|'~' \|\|boam.attr_ext_code | This expression defines how to join the W_DOMAIN_MEMBER_LKP_TL row to the W_RTL_ITEM_GRP1_D row. This value is joinable to the DOMAIN_MEMBER_CODE column, in W_DOMAIN_MEMBER_LKP_TL. |
| EXCLUDE_FLG | N | A Y/N value to either exclude this attribute (Y) or not (N) from processing. |

### Run Attribute ETL Routine

The attribute loading can be performed by running a shell script in the common/scripts/bin directory of the RSE_HOME. Running rse_cda_etl_load.ksh PRODUCT executes the appropriate ETL routine to copy product attributes. This routine should not only be run during system startup, but should also be scheduled for routine execution as part of a maintenance batch.

### Review Attribute for Grouping Requirements

Once the attribute data is in the ORASE tables, it is possible to review the attribute data (in table RSE_PROD_ATTR) to determine if the attribute requires any special grouping. The CDT application requires attribute values that are grouped in logical buckets of values. For example, when adding a Flavor attribute, many different flavors may be available for the product category. Too many distinct values can affect the ability to detect purchasing patterns for customers, so the data must be reduced to a manageable set of values (no more than 5 values). This process is known as attribute grouping or attribute binning.

Some attributes may also be used to arrive at different attributes. For example, with Flavor, it is possible to arrive at just two groups of Flavored and Non Flavored, but it might also be desired to further distinguish between the different types of Flavored values (such as Non Flavored, Fruit Flavored, Mild Flavored, and Specialty Flavored). This type of processing requires knowledge of the product attributes, so that the values can be correlated correctly. For this example, the single attribute that is added for Flavor is introduced as two separate groups of attributes, as just described.

### Populate RSE_PROD_ATTR_GRP_VALUE_STG Interface (Attribute Value Groups)

Once the attribute data has been reviewed and groups have been defined, it is necessary to define the attribute group and process them into the database. The output of the prior step must be loaded into ORASE's 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 9–3 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<br>CLS~1000~10000~flavor_type | This must be a unique value to describe the attribute to be used by ORASE modules. Since the source Flavor attribute is being defined as two different attributes for ORASE, two example values are shown here. |
| PROD_ATTR_GRP_NAME | FlavorYN<br>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 Indentifier<br>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. |

*Table 9–3 (Cont.) RSE_PROD_ATTR_GRP_VALUE_STG*

| Column | Example | Description |
|---|---|---|
| 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).<br>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 9–4 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 |
| 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 (Attribute Value Group Cross Reference)

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 9–5 RSE_PROD_ATTR_VALUE_XREF_STG*

| Column | Example | Description |
|---|---|---|
| PROD_ATTR_VALUE_KEY | CLS~1000~10000~flavor_yn~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. |
| ATTR_VALUE_EXT_CODE | 32 | 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 of some examples for adding a new flavor attribute, using string based attributes.

*Table 9–6 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 |

*Table 9–6 (Cont.) Adding a New Flavor Attribute*

| PROD_ATTR_VALUE_KEY | ATTR_STRING_VALUE |
|---|---|
| CLS~1000~10000~flavor_type~special | CINNAMON |
| CLS~1000~10000~flavor_type~special | CARAMEL |

## Process Attribute Group Interfaces

Once the staging tables have been populated with data, it is time to process these interfaces and load them into the target tables.

### Execute Product Attribute Group Value Interface

The data for RSE_PROD_ATTR_GRP_VALUE_STG can either be directly populated into the database or it can be loaded into the database from a text file, using scripts provided by the application.

In $RSE_HOME/common/scripts/bin, a script called rse_prod_attr_grp_value_stg.ksh is available to load a data file located on the Linux server. A directory called $RSE_HOME/common/data/infile is available for storing these types of inbound data files. Once the file is available to be loaded from the Linux server, the rse_prod_attr_grp_value_stg.ksh script should be executed, passing it the full path name of the file to be loaded.

Regardless of whether the staging table was populated via the script referred to in the previous paragraph, or directly loaded, once it is loaded and ready for processing, the script rse_prod_attr_grp_value_load.ksh (located at $RSE_HOME/common/scripts/bin/) can be executed.

If there are any invalid data records, then the table RSE_PROD_ATTR_GRP_VALUE_BAD will be populated with the rows that failed to pass validation. These rows are populated with four standard columns for an interface populated via this method. These columns are ERROR_ROWID, ERROR_ID, ERROR_DESCR, and ERROR_DT. These columns provide insight into the problems with the interface data. Once the data is corrected and re-staged in the staging tables, it can be processed again.

### Execute Product Attribute Value Cross Reference Interface

The data for RSE_PROD_ATTR_VALUE_XREF_STG can either be directly populated into the database or it can be loaded into the database from a text file, using scripts provided by the application.

In $RSE_HOME/common/scripts/bin, a script called rse_prod_attr_value_xref_stg.ksh is available to load a data file located on the Linux server. A directory called $RSE_HOME/common/data/infile is available for storing these types of inbound data files. Once the file is available to be loaded from the Linux server, the rse_prod_attr_value_xref_stg.ksh script should be executed, passing it the full path name of the file to be loaded.

Regardless of whether the staging table was populated via the script referred to in the previous paragraph, or directly loaded, once it is loaded and ready for processing, the script rse_prod_attr_value_xref_load.ksh (located at $RSE_HOME/common/scripts/bin/) can be executed.

If there are any invalid data records, then the table RSE_PROD_ATTR_VALUE_XREF_BAD will be populated with the rows that failed to pass validation. These rows are populated with four standard columns for an interface populated via this method. These columns are ERROR_ROWID, ERROR_ID, ERROR_DESCR, and ERROR_DT.

These columns provide insight into the problems with the interface data. Once the data is corrected and re-staged in the staging tables, it can be processed again.

## Post Processing

After new product attributes and their attribute groups have been defined, there are some standard processes that should be executed, depending on the modules being used from ORASE.

Advanced Clustering (AC) can use product attributes as a clustering criteria. AC uses what is known as attribute share, and for this, it requires some aggregate data to be available, and it also requires attributes to be defined as criteria within the AC metadata tables.

AC has a configuration in RSE_CONFIG called PERF_CIS_APPROACH. This configuration can have a value of CDT if it should use the same types of attribute data as the CDT application uses (RSE_PROD_ATTR_VALUE_XREF_STG rows), or it can have a value of DT to match the attribute data that the DT application uses (base attribute data).

### Define AC Product Attribute Metadata

The first step required for AC is to execute the cis_prod_attr_maint.ksh script in $CIS_HOME/scripts/bin. This routine defines the attributes and their values in the appropriate AC metadata tables.

### Update Aggregate Attribute Sales

A batch process is provided in the common directory to calculate aggregate sales data for product attributes. In $RSE_HOME/common/scripts/bin, a script is available called rse_wkly_sls_ph_attr_aggr_setup.ksh that is responsible for performing an update of weekly product attribute sales aggregations. The script accepts three parameters that can be used for routine maintenance, required when performing large scale attribute definitions. The first optional parameter refers to the number of weeks of data that should be updated when the process runs. The second parameter is a Y/N flag to signify whether the process should force the updating of weeks that have previously been completed (Y) or not (N: the default). The last parameter is an ID that represents the maximum day for which the data should be updated. In most situations, this last parameter should not be required.

After the appropriate work has been queued for execution, the processes can be executed by running the script called rse_wkly_sls_process.ksh in the same directory.

### Calculate Attribute Sales Share

The AC product attribute process uses store share metrics for its clustering algorithm. This process requires that store shares be calculated for the attributes used by AC. In the $RSE_HOME/cis/scripts/bin directory, there are two scripts that manage this processing. These routines are expected to be run as part of routine weekly maintenance, but can also be run to refresh prior weeks if significant attribute maintenance has been performed.

The first script to be executed is cis_prod_attr_loc_share_setup.ksh, and it accommodates three optional parameters. The first optional parameter refers to the number of weeks of data that should be updated when the process runs. The second parameter is a Y/N flag to signify whether the process should force the updating of weeks that have previously been completed (Y) or not (N: the default). The last parameter is an ID that represents the maximum day for which the data should be updated. In most situations, this last parameter should not be required.

The second script to be executed is cis_prod_attr_loc_share_process.ksh. This script performs the processing of the weeks that were queued for execution by the prior script.

## Attribute Maintenance Completed

At the completion of the Calculate Attribute Sales Share step, all processing that is required when major attribute maintenance has been performed has been completed and the various applications should be able to use the newly provided data.

# A

# Sample Taskflow_MultiSolution.xml File for ASO/Fusion Client Integration

Sample Taskflow_MultiSolution.xml file. The ASO Module entries in the file begin with <module_step>. Note there are two and the <order> value is important.

```
<?xml version="1.0" encoding="UTF-8"?>
<activities xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.example.org/App_Navigation.xsd"
 xmlns="http://www.example.org">
  <activity_group>
    <name>catman.AG_catman</name>
    <description>catman.AG_catman.Desc</description>
    <order_num>1</order_num>
    <activity>
      <name>catman.AG_catman.Activity1</name>
      <description>catman.AG_catman.Activity1.Desc</description>
      <order_num>1</order_num>
      <task>
        <name>catman.AG_catman.Activity1.Task4</name>
        <description>catman.AG_catman.Activity1.Task4.Desc</description>
        <solution>catman</solution>
        <order_num>1</order_num>
        <hide>false</hide>
        <dynamic>false</dynamic>
        <wizard_only>false</wizard_only>
        <unassigned_worksheets>false</unassigned_worksheets>
        <wkbk_template>
          <wkbk_template_grp>Planning</wkbk_template_grp>
          <wkbk_template_name>CatPlanning</wkbk_template_name>
        </wkbk_template>
        <step>
          <name>catman.AG_catman.Activity1.Task4.Step11</name>
          <description>catman.AG_catman.Activity1.Task4.Step11.Desc</description>
          <order_num>1</order_num>
          <instructions>catman.AG_
catman.Activity1.Task4.Step11.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_sa_01" >
            <name>cm_ws_sa_01</name>
            <tab>catman.AG_catman.Activity1.Task4.Step11.Tab34</tab>
          </worksheet>
        </step>
        <step>
          <name>catman.AG_catman.Activity1.Task4.Step12</name>
          <description>catman.AG_catman.Activity1.Task4.Step12.Desc</description>
          <order_num>2</order_num>
          <instructions>catman.AG_
```

```
catman.Activity1.Task4.Step12.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_rs_01" >
            <name>cm_ws_rs_01</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_rs_02" >
            <name>cm_ws_rs_02</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_rs_03" >
            <name>cm_ws_rs_03</name>
          </worksheet>
        </step>
        <step>
          <name>catman.AG_catman.Activity1.Task4.Step13</name>
          <description>catman.AG_catman.Activity1.Task4.Step13.Desc</description>
          <order_num>3</order_num>
          <instructions>catman.AG_
catman.Activity1.Task4.Step13.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_01" >
            <name>cm_ws_cr_01</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab35</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_02" >
            <name>cm_ws_cr_02</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab35</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_03" >
            <name>cm_ws_cr_03</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab35</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_04" >
            <name>cm_ws_cr_04</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab35</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_05" >
            <name>cm_ws_cr_05</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab36</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_06" >
            <name>cm_ws_cr_06</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab36</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_07" >
            <name>cm_ws_cr_07</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab36</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_08" >
            <name>cm_ws_cr_08</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab36</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_09" >
            <name>cm_ws_cr_09</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab36</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_10" >
            <name>cm_ws_cr_10</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab37</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_11" >
            <name>cm_ws_cr_11</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab37</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_12" >
            <name>cm_ws_cr_12</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab37</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_13" >
            <name>cm_ws_cr_13</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab37</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_14" >
            <name>cm_ws_cr_14</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab38</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_15" >
            <name>cm_ws_cr_15</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab38</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_16" >
            <name>cm_ws_cr_16</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab38</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_17" >
            <name>cm_ws_cr_17</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab39</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_18" >
            <name>cm_ws_cr_18</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab39</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cr_19" >
            <name>cm_ws_cr_19</name>
            <tab>catman.AG_catman.Activity1.Task4.Step13.Tab39</tab>
          </worksheet>
        </step>
        <step>
          <name>catman.AG_catman.Activity1.Task4.Step14</name>
          <description>catman.AG_catman.Activity1.Task4.Step14.Desc</description>
          <order_num>4</order_num>
          <instructions>catman.AG_
catman.Activity1.Task4.Step14.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cs_01" >
            <name>cm_ws_cs_01</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cs_02" >
            <name>cm_ws_cs_02</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cs_03" >
            <name>cm_ws_cs_03</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cs_04" >
            <name>cm_ws_cs_04</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cs_05" >
            <name>cm_ws_cs_05</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cs_06" >
            <name>cm_ws_cs_06</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cs_07" >
            <name>cm_ws_cs_07</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_cs_08" >
            <name>cm_ws_cs_08</name>
          </worksheet>
        </step>
        <step>
          <name>catman.AG_catman.Activity1.Task4.Step15</name>
          <description>catman.AG_catman.Activity1.Task4.Step15.Desc</description>
          <order_num>5</order_num>
          <instructions>catman.AG_
catman.Activity1.Task4.Step15.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_ct_01" >
            <name>cm_ws_ct_01</name>
            <tab>catman.AG_catman.Activity1.Task4.Step15.Tab40</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_ct_02" >
            <name>cm_ws_ct_02</name>
            <tab>catman.AG_catman.Activity1.Task4.Step15.Tab40</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_ct_03" >
            <name>cm_ws_ct_03</name>
            <tab>catman.AG_catman.Activity1.Task4.Step15.Tab40</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_ct_04" >
            <name>cm_ws_ct_04</name>
            <tab>catman.AG_catman.Activity1.Task4.Step15.Tab40</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_ct_06" >
            <name>cm_ws_ct_06</name>
            <tab>catman.AG_catman.Activity1.Task4.Step15.Tab41</tab>
          </worksheet>
        </step>
        <step>
          <name>catman.AG_catman.Activity1.Task4.Step16</name>
          <description>catman.AG_catman.Activity1.Task4.Step16.Desc</description>
          <order_num>6</order_num>
          <instructions>catman.AG_
catman.Activity1.Task4.Step16.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_sa_01" >
            <name>cm_ws_sa_01</name>
            <tab>catman.AG_catman.Activity1.Task4.Step16.Tab42</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.worksheet4" >
            <name>worksheet4</name>
            <tab>catman.AG_catman.Activity1.Task4.Step16.Tab44</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.worksheet3" >
            <name>worksheet3</name>
            <tab>catman.AG_catman.Activity1.Task4.Step16.Tab44</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_sa_02" >
            <name>cm_ws_sa_02</name>
            <tab>catman.AG_catman.Activity1.Task4.Step16.Tab43</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_sa_03" >
            <name>cm_ws_sa_03</name>
            <tab>catman.AG_catman.Activity1.Task4.Step16.Tab43</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_sa_04" >
            <name>cm_ws_sa_04</name>
            <tab>catman.AG_catman.Activity1.Task4.Step16.Tab43</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_sa_05" >
            <name>cm_ws_sa_05</name>
            <tab>catman.AG_catman.Activity1.Task4.Step16.Tab45</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task4.cm_ws_sa_06" >
            <name>cm_ws_sa_06</name>
            <tab>catman.AG_catman.Activity1.Task4.Step16.Tab46</tab>
          </worksheet>
          <custom_menu label="catman.AG_catman.Activity1.Task4.Step16.menu0"
>1484</custom_menu>
          <custom_menu label="catman.AG_
catman.Activity1.Task4.Step16.menuitem1469" >1469</custom_menu>
        </step>
      </task>
      <task>
        <name>catman.AG_catman.Activity1.Task1</name>
        <description>catman.AG_catman.Activity1.Task1.Desc</description>
        <solution>catman</solution>
        <order_num>2</order_num>
        <hide>false</hide>
        <dynamic>false</dynamic>
        <wizard_only>false</wizard_only>
        <unassigned_worksheets>false</unassigned_worksheets>
        <wkbk_template>
          <wkbk_template_grp>Planning</wkbk_template_grp>
          <wkbk_template_name>ARCluster</wkbk_template_name>
        </wkbk_template>
        <step>
          <name>catman.AG_catman.Activity1.Task1.Step1</name>
          <description>catman.AG_catman.Activity1.Task1.Step1.Desc</description>
          <order_num>1</order_num>
          <instructions>catman.AG_
catman.Activity1.Task1.Step1.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab4_ws_1" >
            <name>ac_tab4_ws_1</name>
            <tab>catman.AG_catman.Activity1.Task1.Step1.Tab1</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab4_ws_2" >
            <name>ac_tab4_ws_2</name>
            <tab>catman.AG_catman.Activity1.Task1.Step1.Tab2</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab4_ws_3" >
            <name>ac_tab4_ws_3</name>
            <tab>catman.AG_catman.Activity1.Task1.Step1.Tab2</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab4_ws_4" >
            <name>ac_tab4_ws_4</name>
            <tab>catman.AG_catman.Activity1.Task1.Step1.Tab2</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab4_ws_5" >
            <name>ac_tab4_ws_5</name>
            <tab>catman.AG_catman.Activity1.Task1.Step1.Tab2</tab>
          </worksheet>
        </step>
        <step>
          <name>catman.AG_catman.Activity1.Task1.Step2</name>
          <description>catman.AG_catman.Activity1.Task1.Step2.Desc</description>
          <order_num>2</order_num>
          <instructions>catman.AG_
catman.Activity1.Task1.Step2.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_1" >
            <name>ac_ws_ma_1</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab3</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_2" >
            <name>ac_ws_ma_2</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab3</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_3" >
            <name>ac_ws_ma_3</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab4</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_6" >
            <name>ac_ws_ma_6</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab4</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_4" >
            <name>ac_ws_ma_4</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab5</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_7" >
            <name>ac_ws_ma_7</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab5</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_16" >
            <name>ac_ws_ma_16</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab6</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_17" >
            <name>ac_ws_ma_17</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab6</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_5" >
            <name>ac_ws_ma_5</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab7</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_8" >
            <name>ac_ws_ma_8</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab7</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_11" >
            <name>ac_ws_ma_11</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab8</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_12" >
            <name>ac_ws_ma_12</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab8</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_15" >
            <name>ac_ws_ma_15</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab8</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_13" >
            <name>ac_ws_ma_13</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab9</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_ws_ma_14" >
            <name>ac_ws_ma_14</name>
            <tab>catman.AG_catman.Activity1.Task1.Step2.Tab9</tab>
          </worksheet>
          <custom_menu label="catman.AG_catman.Activity1.Task1.Step2.menuitem114"
>114</custom_menu>
        </step>
        <step>
          <name>catman.AG_catman.Activity1.Task1.Step3</name>
          <description>catman.AG_catman.Activity1.Task1.Step3.Desc</description>
          <order_num>3</order_num>
          <instructions>catman.AG_
catman.Activity1.Task1.Step3.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab6_ws_1" >
            <name>ac_tab6_ws_1</name>
            <tab>catman.AG_catman.Activity1.Task1.Step3.Tab10</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab6_ws_2" >
            <name>ac_tab6_ws_2</name>
            <tab>catman.AG_catman.Activity1.Task1.Step3.Tab11</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab6_ws_3" >
            <name>ac_tab6_ws_3</name>
            <tab>catman.AG_catman.Activity1.Task1.Step3.Tab11</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab6_ws_4" >
            <name>ac_tab6_ws_4</name>
            <tab>catman.AG_catman.Activity1.Task1.Step3.Tab11</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab6_ws_5" >
            <name>ac_tab6_ws_5</name>
            <tab>catman.AG_catman.Activity1.Task1.Step3.Tab12</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab6_ws_6" >
            <name>ac_tab6_ws_6</name>
            <tab>catman.AG_catman.Activity1.Task1.Step3.Tab12</tab>
          </worksheet>
        </step>
        <step>
          <name>catman.AG_catman.Activity1.Task1.Step4</name>
          <description>catman.AG_catman.Activity1.Task1.Step4.Desc</description>
          <order_num>4</order_num>
          <instructions>catman.AG_
catman.Activity1.Task1.Step4.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab9_ws1" >
            <name>ac_tab9_ws1</name>
            <tab>catman.AG_catman.Activity1.Task1.Step4.Tab13</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab9_ws2" >
            <name>ac_tab9_ws2</name>
            <tab>catman.AG_catman.Activity1.Task1.Step4.Tab13</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab9_ws3" >
            <name>ac_tab9_ws3</name>
            <tab>catman.AG_catman.Activity1.Task1.Step4.Tab13</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab9_ws4" >
            <name>ac_tab9_ws4</name>
            <tab>catman.AG_catman.Activity1.Task1.Step4.Tab14</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task1.ac_tab1_ws_5" >
            <name>ac_tab1_ws_5</name>
            <tab>catman.AG_catman.Activity1.Task1.Step4.Tab15</tab>
          </worksheet>
        </step>
      </task>
      <task>
        <name>catman.AG_catman.Activity1.Task2</name>
        <description>catman.AG_catman.Activity1.Task2.Desc</description>
        <solution>catman</solution>
        <order_num>3</order_num>
        <hide>false</hide>
        <dynamic>false</dynamic>
        <wizard_only>false</wizard_only>
        <unassigned_worksheets>false</unassigned_worksheets>
        <wkbk_template>
          <wkbk_template_grp>Planning</wkbk_template_grp>
          <wkbk_template_name>ARCluster</wkbk_template_name>
        </wkbk_template>
        <step>
          <name>catman.AG_catman.Activity1.Task2.Step5</name>
          <description>catman.AG_catman.Activity1.Task2.Step5.Desc</description>
          <order_num>1</order_num>
          <instructions>catman.AG_
catman.Activity1.Task2.Step5.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_ws_as_1" >
            <name>ac_ws_as_1</name>
            <tab>catman.AG_catman.Activity1.Task2.Step5.Tab16</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_ws_as_2" >
            <name>ac_ws_as_2</name>
            <tab>catman.AG_catman.Activity1.Task2.Step5.Tab16</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab1_ws_8" >
            <name>ac_tab1_ws_8</name>
            <tab>catman.AG_catman.Activity1.Task2.Step5.Tab47</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_ws_as_3" >
            <name>ac_ws_as_3</name>
            <tab>catman.AG_catman.Activity1.Task2.Step5.Tab17</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_ws_as_4" >
            <name>ac_ws_as_4</name>
            <tab>catman.AG_catman.Activity1.Task2.Step5.Tab17</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab1_ws_7" >
            <name>ac_tab1_ws_7</name>
            <tab>catman.AG_catman.Activity1.Task2.Step5.Tab17</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab1_ws_9" >
            <name>ac_tab1_ws_9</name>
            <tab>catman.AG_catman.Activity1.Task2.Step5.Tab17</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab1_ws_5" >
            <name>ac_tab1_ws_5</name>
            <tab>catman.AG_catman.Activity1.Task2.Step5.Tab18</tab>
          </worksheet>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step5.menuitem112"
>112</custom_menu>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step5.menuitem113"
>113</custom_menu>
        </step>
        <step>
          <name>catman.AG_catman.Activity1.Task2.Step6</name>
          <description>catman.AG_catman.Activity1.Task2.Step6.Desc</description>
          <order_num>2</order_num>
          <instructions>catman.AG_
catman.Activity1.Task2.Step6.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab5_ws_1" >
            <name>ac_tab5_ws_1</name>
            <tab>catman.AG_catman.Activity1.Task2.Step6.Tab19</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab5_ws_2" >
            <name>ac_tab5_ws_2</name>
            <tab>catman.AG_catman.Activity1.Task2.Step6.Tab19</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab5_ws_3" >
            <name>ac_tab5_ws_3</name>
            <tab>catman.AG_catman.Activity1.Task2.Step6.Tab19</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab5_ws_4" >
            <name>ac_tab5_ws_4</name>
            <tab>catman.AG_catman.Activity1.Task2.Step6.Tab19</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab5_ws_5" >
            <name>ac_tab5_ws_5</name>
            <tab>catman.AG_catman.Activity1.Task2.Step6.Tab19</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab5_ws_9" >
            <name>ac_tab5_ws_9</name>
            <tab>catman.AG_catman.Activity1.Task2.Step6.Tab21</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab5_ws_10" >
            <name>ac_tab5_ws_10</name>
            <tab>catman.AG_catman.Activity1.Task2.Step6.Tab21</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab5_ws_11" >
            <name>ac_tab5_ws_11</name>
            <tab>catman.AG_catman.Activity1.Task2.Step6.Tab21</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab5_ws_6" >
            <name>ac_tab5_ws_6</name>
            <tab>catman.AG_catman.Activity1.Task2.Step6.Tab20</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab5_ws_7" >
            <name>ac_tab5_ws_7</name>
            <tab>catman.AG_catman.Activity1.Task2.Step6.Tab22</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_ipi_ws_1" >
            <name>ac_ipi_ws_1</name>
            <tab>catman.AG_catman.Activity1.Task2.Step6.Tab22</tab>
          </worksheet>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step6.menuitem1472"
>1472</custom_menu>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step6.menuitem1475"
>1475</custom_menu>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step6.menuitem1476"
>1476</custom_menu>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step6.menuitem1492"
>1492</custom_menu>
        </step>
        <step>
```

```
          <name>catman.AG_catman.Activity1.Task2.Step7</name>
          <description>catman.AG_catman.Activity1.Task2.Step7.Desc</description>
          <order_num>3</order_num>
          <instructions>catman.AG_
catman.Activity1.Task2.Step7.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_ws_cmca_1" >
            <name>ac_ws_cmca_1</name>
            <tab>catman.AG_catman.Activity1.Task2.Step7.Tab23</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_ws_cmca_2" >
            <name>ac_ws_cmca_2</name>
            <tab>catman.AG_catman.Activity1.Task2.Step7.Tab23</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_ws_cmca_3" >
            <name>ac_ws_cmca_3</name>
            <tab>catman.AG_catman.Activity1.Task2.Step7.Tab33</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.worksheet8" >
            <name>worksheet8</name>
            <tab>catman.AG_catman.Activity1.Task2.Step7.Tab33</tab>
          </worksheet>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step7.menuitem1470"
>1470</custom_menu>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step7.menuitem1493"
>1493</custom_menu>
        </step>
        <step>
          <name>catman.AG_catman.Activity1.Task2.Step31</name>
          <description>catman.AG_catman.Activity1.Task2.Step31.Desc</description>
          <order_num>4</order_num>
          <instructions>catman.AG_
catman.Activity1.Task2.Step31.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task2.ic_tab1_ws1" >
            <name>ic_tab1_ws1</name>
            <tab>catman.AG_catman.Activity1.Task2.Step31.Tab50</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ic_tab1_ws2" >
            <name>ic_tab1_ws2</name>
            <tab>catman.AG_catman.Activity1.Task2.Step31.Tab50</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ic_tab1_ws3" >
            <name>ic_tab1_ws3</name>
            <tab>catman.AG_catman.Activity1.Task2.Step31.Tab51</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ic_tab1_ws4" >
            <name>ic_tab1_ws4</name>
            <tab>catman.AG_catman.Activity1.Task2.Step31.Tab51</tab>
          </worksheet>
          <custom_menu label="catman.AG_
catman.Activity1.Task2.Step31.menuitem1495" >1495</custom_menu>
        </step>
        <step>
          <name>catman.AG_catman.Activity1.Task2.Step8</name>
          <description>catman.AG_catman.Activity1.Task2.Step8.Desc</description>
          <order_num>5</order_num>
          <instructions>catman.AG_
catman.Activity1.Task2.Step8.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab8_ws_1" >
            <name>ac_tab8_ws_1</name>
            <tab>catman.AG_catman.Activity1.Task2.Step8.Tab24</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.worksheet9" >
            <name>worksheet9</name>
            <tab>catman.AG_catman.Activity1.Task2.Step8.Tab24</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab8_ws_2" >
            <name>ac_tab8_ws_2</name>
            <tab>catman.AG_catman.Activity1.Task2.Step8.Tab25</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab11_ws1" >
            <name>ac_tab11_ws1</name>
            <tab>catman.AG_catman.Activity1.Task2.Step8.Tab48</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab11_ws2" >
            <name>ac_tab11_ws2</name>
            <tab>catman.AG_catman.Activity1.Task2.Step8.Tab48</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task2.ac_tab7_ws1" >
            <name>ac_tab7_ws1</name>
            <tab>catman.AG_catman.Activity1.Task2.Step8.Tab26</tab>
          </worksheet>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step8.menuitem1477"
>1477</custom_menu>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step8.menuitem1471"
>1471</custom_menu>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step8.menuitem1494"
>1494</custom_menu>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step8.menuitem1473"
>1473</custom_menu>
          <custom_menu label="catman.AG_catman.Activity1.Task2.Step8.menuitem1489"
>1489</custom_menu>
        </step>

        <module_step>
          <name>sia-so.asomain.microapp.name</name>
          <description>sia-so.asomain.microapp.desc</description>
          <module>asomain</module>
          <module_bundle>sia-so</module_bundle>
          <order_num>6</order_num>
      </module_step>

      </task>
      <task>
        <name>catman.AG_catman.Activity1.Task3</name>
        <description>catman.AG_catman.Activity1.Task3.Desc</description>
        <solution>catman</solution>
        <order_num>4</order_num>
        <hide>false</hide>
        <dynamic>false</dynamic>
        <wizard_only>false</wizard_only>
        <unassigned_worksheets>false</unassigned_worksheets>
        <wkbk_template>
          <wkbk_template_grp>Planning</wkbk_template_grp>
          <wkbk_template_name>ARCluster</wkbk_template_name>
        </wkbk_template>
        <step>
          <name>catman.AG_catman.Activity1.Task3.Step9</name>
          <description>catman.AG_catman.Activity1.Task3.Step9.Desc</description>
          <order_num>1</order_num>
          <instructions>catman.AG_
```

```
catman.Activity1.Task3.Step9.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab10_ws1" >
            <name>ac_tab10_ws1</name>
            <tab>catman.AG_catman.Activity1.Task3.Step9.Tab27</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab10_ws2" >
            <name>ac_tab10_ws2</name>
            <tab>catman.AG_catman.Activity1.Task3.Step9.Tab27</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab10_ws3" >
            <name>ac_tab10_ws3</name>
            <tab>catman.AG_catman.Activity1.Task3.Step9.Tab27</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab10_ws4" >
            <name>ac_tab10_ws4</name>
            <tab>catman.AG_catman.Activity1.Task3.Step9.Tab27</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab10_ws5" >
            <name>ac_tab10_ws5</name>
            <tab>catman.AG_catman.Activity1.Task3.Step9.Tab27</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab10_ws7" >
            <name>ac_tab10_ws7</name>
            <tab>catman.AG_catman.Activity1.Task3.Step9.Tab28</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab10_ws8" >
            <name>ac_tab10_ws8</name>
            <tab>catman.AG_catman.Activity1.Task3.Step9.Tab28</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab10_ws9" >
            <name>ac_tab10_ws9</name>
            <tab>catman.AG_catman.Activity1.Task3.Step9.Tab28</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab10_ws10" >
            <name>ac_tab10_ws10</name>
            <tab>catman.AG_catman.Activity1.Task3.Step9.Tab29</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab10_ws11" >
            <name>ac_tab10_ws11</name>
            <tab>catman.AG_catman.Activity1.Task3.Step9.Tab30</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.as_dt_details_1" >
            <name>as_dt_details_1</name>
            <tab>catman.AG_catman.Activity1.Task3.Step9.Tab30</tab>
          </worksheet>
          <custom_menu label="catman.AG_catman.Activity1.Task3.Step9.menuitem1485"
>1485</custom_menu>
          <custom_menu label="catman.AG_catman.Activity1.Task3.Step9.menuitem1486"
>1486</custom_menu>
          <custom_menu label="catman.AG_catman.Activity1.Task3.Step9.menuitem1487"
>1487</custom_menu>
          <custom_menu label="catman.AG_catman.Activity1.Task3.Step9.menuitem1497"
>1497</custom_menu>
        </step>
        <step>
          <name>catman.AG_catman.Activity1.Task3.Step10</name>
          <description>catman.AG_catman.Activity1.Task3.Step10.Desc</description>
          <order_num>2</order_num>
          <instructions>catman.AG_
catman.Activity1.Task3.Step10.Instructions</instructions>
```

```
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab10_ws13" >
            <name>ac_tab10_ws13</name>
            <tab>catman.AG_catman.Activity1.Task3.Step10.Tab31</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab11_ws1" >
            <name>ac_tab11_ws1</name>
            <tab>catman.AG_catman.Activity1.Task3.Step10.Tab49</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab11_ws2" >
            <name>ac_tab11_ws2</name>
            <tab>catman.AG_catman.Activity1.Task3.Step10.Tab49</tab>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity1.Task3.ac_tab10_ws12" >
            <name>ac_tab10_ws12</name>
            <tab>catman.AG_catman.Activity1.Task3.Step10.Tab32</tab>
          </worksheet>
          <custom_menu label="catman.AG_
catman.Activity1.Task3.Step10.menuitem1488" >1488</custom_menu>
          <custom_menu label="catman.AG_
catman.Activity1.Task3.Step10.menuitem1490" >1490</custom_menu>
        </step>

      <module_step>
        <name>sia-so.asomain.microapp.name</name>
        <description>sia-so.asomain.microapp.desc</description>
        <module>asomain</module>
        <module_bundle>sia-so</module_bundle>
        <order_num>2</order_num>
      </module_step>

     </task>
    </activity>
    <activity>
      <name>catman.AG_catman.Activity2</name>
      <description>catman.AG_catman.Activity2.Desc</description>
      <order_num>2</order_num>
      <task>
        <name>catman.AG_catman.Activity2.Task5</name>
        <description>catman.AG_catman.Activity2.Task5.Desc</description>
        <solution>catman</solution>
        <order_num>1</order_num>
        <hide>false</hide>
        <dynamic>false</dynamic>
        <wizard_only>false</wizard_only>
        <unassigned_worksheets>false</unassigned_worksheets>
        <wkbk_template>
          <wkbk_template_grp>CatManAdmin</wkbk_template_grp>
          <wkbk_template_name>CatManAdmin</wkbk_template_name>
        </wkbk_template>
        <step>
          <name>catman.AG_catman.Activity2.Task5.Step17</name>
          <description>catman.AG_catman.Activity2.Task5.Step17.Desc</description>
          <order_num>1</order_num>
          <instructions>catman.AG_
catman.Activity2.Task5.Step17.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity2.Task5.worksheet2" >
            <name>worksheet2</name>
          </worksheet>
        </step>
        <step>
```

```
          <name>catman.AG_catman.Activity2.Task5.Step28</name>
          <description>catman.AG_catman.Activity2.Task5.Step28.Desc</description>
          <order_num>2</order_num>
          <instructions>catman.AG_
catman.Activity2.Task5.Step28.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity2.Task5.cm_admin_ws_6" >
            <name>cm_admin_ws_6</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity2.Task5.cm_admin_ws_10" >
            <name>cm_admin_ws_10</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity2.Task5.worksheet1" >
            <name>worksheet1</name>
          </worksheet>
          <custom_menu label="catman.AG_
catman.Activity2.Task5.Step28.menuitem1491" >1491</custom_menu>
        </step>
        <step>
          <name>catman.AG_catman.Activity2.Task5.Step19</name>
          <description>catman.AG_catman.Activity2.Task5.Step19.Desc</description>
          <order_num>3</order_num>
          <instructions>catman.AG_
catman.Activity2.Task5.Step19.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity2.Task5.cm_admin_ws_1" >
            <name>cm_admin_ws_1</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity2.Task5.cm_admin_ws_2" >
            <name>cm_admin_ws_2</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity2.Task5.cm_admin_ws_3" >
            <name>cm_admin_ws_3</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity2.Task5.cm_admin_ws_4" >
            <name>cm_admin_ws_4</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity2.Task5.cm_admin_11" >
            <name>cm_admin_11</name>
          </worksheet>
        </step>
        <step>
          <name>catman.AG_catman.Activity2.Task5.Step21</name>
          <description>catman.AG_catman.Activity2.Task5.Step21.Desc</description>
          <order_num>4</order_num>
          <instructions>catman.AG_
catman.Activity2.Task5.Step21.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity2.Task5.cm_admin_ws_5" >
            <name>cm_admin_ws_5</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity2.Task5.cm_admin_ws_9" >
            <name>cm_admin_ws_9</name>
          </worksheet>
        </step>
        <step>
          <name>catman.AG_catman.Activity2.Task5.Step24</name>
          <description>catman.AG_catman.Activity2.Task5.Step24.Desc</description>
          <order_num>5</order_num>
          <instructions>catman.AG_
catman.Activity2.Task5.Step24.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity2.Task5.cm_admin_ws_at" >
            <name>cm_admin_ws_at</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity2.Task5.worksheet6" >
            <name>worksheet6</name>
          </worksheet>
        </step>
        <step>
          <name>catman.AG_catman.Activity2.Task5.Step25</name>
          <description>catman.AG_catman.Activity2.Task5.Step25.Desc</description>
          <order_num>6</order_num>
          <instructions>catman.AG_
catman.Activity2.Task5.Step25.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity2.Task5.cm_admin_ws_7" >
            <name>cm_admin_ws_7</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity2.Task5.cm_admin_ws_8" >
            <name>cm_admin_ws_8</name>
          </worksheet>
        </step>
        <step>
          <name>catman.AG_catman.Activity2.Task5.Step30</name>
          <description>catman.AG_catman.Activity2.Task5.Step30.Desc</description>
          <order_num>7</order_num>
          <instructions>catman.AG_
catman.Activity2.Task5.Step30.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity2.Task5.worksheet7" >
            <name>worksheet7</name>
          </worksheet>
        </step>
      </task>
      <task>
        <name>catman.AG_catman.Activity2.Task6</name>
        <description>catman.AG_catman.Activity2.Task6.Desc</description>
        <solution>catman</solution>
        <order_num>2</order_num>
        <hide>false</hide>
        <dynamic>false</dynamic>
        <wizard_only>false</wizard_only>
        <unassigned_worksheets>false</unassigned_worksheets>
        <wkbk_template>
          <wkbk_template_grp>CDTEditor</wkbk_template_grp>
          <wkbk_template_name>CDTE</wkbk_template_name>
        </wkbk_template>
        <step>
          <name>catman.AG_catman.Activity2.Task6.Step27</name>
          <description>catman.AG_catman.Activity2.Task6.Step27.Desc</description>
          <order_num>1</order_num>
          <instructions>catman.AG_
catman.Activity2.Task6.Step27.Instructions</instructions>
          <worksheet label="catman.AG_catman.Activity2.Task6.cdte_adt_vas" >
            <name>cdte_adt_vas</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity2.Task6.cdte_adt_cdti" >
            <name>cdte_adt_cdti</name>
          </worksheet>
          <worksheet label="catman.AG_catman.Activity2.Task6.cdte_adt_cdta" >
            <name>cdte_adt_cdta</name>
          </worksheet>
          <custom_menu label="catman.AG_
catman.Activity2.Task6.Step27.menuitem1474" >1474</custom_menu>
        </step>
```

```
   </task>
  </activity>
 </activity_group>
</activities>
```

# B

# Database Detail Definitions

## Table Definitions

### CDT_APPROVED_EXP_VW

This view provides a list of approved CDTs to the CDT XML export process.

*Table B–1 CDT_APPROVED_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| CM_GRP_ID | NUMBER(10) | The FK for the Category Management group. | N |
| CS_LEAF_ID | VARCHAR2(4000) | The FK for the leaf node of the customer segment hierarchy, which is assigned to this CDT during the export. | Y |
| TA_LEAF_ID | VARCHAR2(4000) | The FK for the location/trade area hierarchy, which is assigned to this CDT during the export. | Y |
| ESC_RANK | NUMBER(10) | The numerical rank to use when escalation of results is required. The lower the number, the more important the rank. | N |
| CS_LEVEL_ID | NUMBER(4) | The hierarchy level number for the customer segment hierarchy. | N |
| CS_ID | NUMBER(10) | The FK for the customer segment hierarchy, which this CDT was created for. | N |
| TA_LEVEL_ID | NUMBER(4) | The hierarchy level number for the trade area/location hierarchy. | N |
| TA_ID | NUMBER(10) | The FK for the trade area/location hierarchy, which this CDT was created for. | N |
| CDT_ID | NUMBER(10) | The FK for the CDT to be exported. | N |
| CDT_VERSION_ID | NUMBER(10) | The FK for the version that this CDT was created under. | Y |
| CDT_VERSION_NAME | VARCHAR2(80) | The name of the version as entered in the CDT UI when this version was created. | Y |
| CREATED_BY_USER | VARCHAR2(255) | The name of the user who created the version and CDTs. | N |
| CREATED_ON_DT | DATE | The date when this CDT was created. | Y |

*Table B–1 (Cont.) CDT_APPROVED_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| CHANGED_ON_DT | DATE | The date when this CDT was last modified. | Y |
| ACTIVE_FLG | VARCHAR2(1) | A flag to indicate whether this CDT is considered active (Y) or not (N) for downstream applications like DT. | Y |
| APPROVED_FLG | VARCHAR2(1) | A flag to indicate whether this CDT is considered approved (Y) or not (N). Only approved CDTs should be exported by this view. | Y |

## CDT_ATTR_VAL_PROD_XREF_EXP_VW

This view provides the data for the Category Management application for export of products and their product attribute values.

*Table B–2 CDT_ATTR_VAL_PROD_XREF_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| MERCHANDISE_ID | VARCHAR2(80) | The external identifier for a product/SKU. | Y |
| ATTRIBUTE_EXTERNAL_ID | VARCHAR2(80) | The external identifier for the category-specific attribute. | N |
| ATTRIBUTE_VALUE_ID | VARCHAR2(80) | The external identifier for the product attribute value. | Y |

## CDT_ATTRIBUTE_EXP_VW

This view provides the complete set of category-specific attributes and their attribute values for an export to the Category Management application.

*Table B–3 CDT_ATTRIBUTE_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ATTRIBUTE_VALUE_ID | VARCHAR2(80) | The external identifier for the attribute value. | Y |
| ATTRIBUTE_VALUE_NAME | VARCHAR2(255) | A descriptive name for the attribute value. | Y |
| ATTRIBUTE_EXTERNAL_ID | VARCHAR2(80) | The external identifier for the category-specific attribute. | N |
| ATTRIBUTE_NAME | VARCHAR2(255) | A descriptive name for the category-specific attribute. | Y |

## CDT_BATCH_INTERVAL_TYPE

This is the lookup table for batch interval types.

*Table B–4 CDT_BATCH_INTERVAL_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| INTERVAL_TYPE | VARCHAR2(1) | D=Default, S=Seasonal, P=Product-specific. | N |
| DESCR | VARCHAR2(40) | Description for the interval type. | Y |

**Notes:**

This table contains the descriptions for the various interval types used within a CDT batch. This object performs no other function other than providing documentation of the different values used in the CDT_BATCH_INTERVAL table.

## CDT_EXCLUDE

This table defines the different types of pruning filters that are available to exclude a CDT from being used during the escalation phase of the CDT workflow.

*Table B–5 CDT_EXCLUDE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(80) | The name of the pruning filter. | N |
| DESCR | VARCHAR2(255) | Descriptive text for the pruning filter. | Y |
| PROC_NAME | VARCHAR2(30) | Name of the process that performs this data filtering. | N |
| DISPLAY_SEQ | NUMBER(10) | The order in which the filters should be displayed when viewing filter statistics in the UI. | N |
| PROCESS_SEQ | NUMBER(10) | The order in which the filters should be executed during the filter stage. | N |
| ENABLE_FLG | VARCHAR2(1) | A Y/N indicator that indicates whether the filter is enabled for use. | Y |

**Notes:**

This table can be used to customize the pruning filters that run after CDTs have been created. This allows for the flexibility in customized rules for pruning CDTs from the list of usable CDTs.

The NAME and DESCR columns are displayed in the UI if the NAME value is not found in the UI's resource bundle.

The PROC_NAME is the name of the procedure that should be executed to perform this pruning task. This routine should be implemented in either the CDT_PRUNE_T type or in a database type that extends this type and then is configured as the type to execute for the CDT_PRUNE database service.

When the pruning stage runs, this table will be referenced and each enabled filter will be executed in ascending order, based on the PROCESS_SEQ column. Therefore, this column should be unique in order to ensure consistency in the order in which the filters are executed.

The DISPLAY_SEQ column allows for displaying the filters in the UI in a different order than the order in which they may be executed. Although setting these two columns with different values might be confusing to the user, it can be done if needed.

The ENABLE_FLG can be used to exclude a filter from processing. This can be helpful if a filter is no longer deemed relevant to run. Setting this column to N will prevent further execution of the filter, although any existing filter statistics about prior executions of the filter will remain.

## CDT_FILTER

This table defines the different types of data filters that are available to filter the sales transaction data used for the CDT calculation.

*Table B–6 CDT_FILTER*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(80) | The name of the data filter. | N |
| DESCR | VARCHAR2(255) | Descriptive text for this filter. | Y |
| SRVC_TYPE | VARCHAR2(30) | Service type for the implementation of the data filter. | N |
| SRVC_NAME | VARCHAR2(30) | Service name for the implementation for this data filter. | N |
| DISPLAY_SEQ | NUMBER(10) | The order in which the filters should be displayed when viewing filter statistics in the UI. | N |
| PROCESS_SEQ | NUMBER(10) | The order in which the filters should be executed during the filter stage. | N |
| ENABLE_FLG | VARCHAR2(1) | A Y/N indicator that indicates whether the filter is enabled for use. | Y |

**Notes:**

This table contains the list of CDT Filters that can be executed against sales transaction data in order to remove outliers or any data that could lead to bad results. Although the UI has a fixed number of prompts in the data filtering stage, you can use this table offers to add or remove any filter in the data filtering workflow.

When the data filtering stage runs, this table will be referenced and each enabled filter will be executed in ascending order, based on the PROCESS_SEQ column. Therefore, this column should be unique in order to ensure consistency in the order in which the filters are executed.

The DISPLAY_SEQ column can be used for displaying the filters in the UI in a different order than the order in which they may be executed. Although setting these two columns with different values might be confusing to the user, it can be done if needed.

The ENABLE_FLG can be used to exclude a filter from processing. This can be helpful if a filter is no longer deemed relevant to run. Setting this column to N will prevent further execution of the filter, although any existing filter statistics about prior executions of the filter will remain.

The SRVC_TYPE and SRVC_NAME columns can be used for the specification of the actual implementation that performs the data filtering task. These implementations should be extensions of the CDT_FILTER_SLS_TXN_T database type. The SRVC_TYPE should be defined in the RSE_SRVC_TYPE table, and the SRVC_NAME should be defined in the RSE_SRVC_CONFIG table. Because of this configuration, this table offers the ability to run additional SKU or Customer filters.

## CDT_SIM_EXP_VW

This view provides the export of product similarities calculated within the CDT application to Category Management.

*Table B–7 CDT_SIM_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROD_EXT_KEY1 | VARCHAR2(80) | The external identifier for one half of the product pair. | Y |
| CUSTSEG_EXT_KEY | VARCHAR2(80) | The external identifier for the customer segment this data relates to. | Y |
| LOC_EXT_KEY | VARCHAR2(80) | The external identifier for the store location this data relates to. | Y |
| PROD_EXT_KEY2 | VARCHAR2(80) | The external identifier for the other half of the product pair. | Y |
| PROD_SIM | NUMBER(22,7) | The measurement of how similar the two products are to each other. The values range between values of 0 (completely dissimilar) to values of 1 (completely similar). | Y |
| EFFECTIVE_DT_FROM | DATE | The date this similarity value became effective. | Y |
| EFFECTIVE_DT_TO | DATE | The date this similarity value is effective until. | Y |

## CDT_VERSION

This table defines a version to collectively group a batch of CDTs that were created for a particular purpose.

*Table B–8 CDT_VERSION*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(80) | The name of the version as entered and displayed in the UI. | Y |
| DESCR | VARCHAR2(255) | Descriptive text for the version. | Y |
| CREATED_ON_DT | DATE | The date when this version was created. | Y |
| CREATED_BY_USER | VARCHAR2(255) | The user who created this version. | N |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether this version is considered deleted (Y) or not (N). | Y |
| EXTERNAL_FLG | VARCHAR2(1) | This flag identifies versions that were externally provided (Y). | Y |
| APPROVED_FLG | VARCHAR2(1) | This flag identifies the one version that contains all the approved CDTs (Y). | Y |
| CHANGED_BY_USER | VARCHAR2(255) | The user who last changed this version. | Y |
| CHANGED_ON_DT | DATE | The date when the version was last changed. | Y |
| MAX_SRC_WK_ID | NUMBER(10) | The maximum week ID of the source data for this version. | Y |

*Table B–8 (Cont.) CDT_VERSION*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| MIN_SRC_WK_ID | NUMBER(10) | The minimum week ID of the source data for this version. | Y |
| NUM_WEEKS | NUMBER(10) | Number of weeks of data used to create this version. | Y |
| PROC_REQ_QUEUE_ID | NUMBER(10) | FK for the Process Request Queue last used to perform version-specific background tasks. | Y |

**Notes:**

This table is the parent object for a collection of calculated or imported CDTs.

There are two fixed versions in this table that represent the CDTs that are Approved CDTs (APPROVED_FLG=Y) and that represent the CDTs that were externally loaded into the application (EXTERNAL_FLG=Y). Only one version can have an APPROVED_FLG=Y, and only one version can have an EXTERNAL_FLG=Y, and they cannot be the same version. These two versions are created as part of a seed data file.

For all other versions, the application maintains that a given version name (NAME) can only be used once per user (CREATED_BY_USER).

The PROC_REQ_QUEUE_ID will only be populated if there is a Job Processor task running that is owned by this version.

## CIS_ALGORITHM

This table holds the possible algorithms used in Clustering.

*Table B–9 CIS_ALGORITHM*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(50) | This is the name of the algorithm that is registered for use (for example, BaNG). | N |
| DESCR | VARCHAR2(200) | Descriptive text for the algorithm. | Y |
| ALG_CODE | VARCHAR2(200) | This identifies the program/procedure or required call that is needed to call the algorithm. More technical information is required to identify and execute every call. | Y |
| CURRENT_FLG | VARCHAR2(1) | A flag to indicate whether this row is the most recent row (Y) or if it is a historical row (N). | Y |
| CREATED_BY_USER | VARCHAR2(255) | The name of the user who created the record. | N |
| CREATED_ON_DT | TIMESTAMP(6) | The date when this record was created. | N |
| CREATED_BY_USER | VARCHAR2(255) | The name of the user who last changed the record. | Y |
| CHANGED_ON_DT | TIMESTAMP(6) | The date when this record was last changed. | Y |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether the row is considered deleted (Y) or not (N). | Y |

**Notes:**

This table holds the possible algorithms used in Clustering.

## CIS_ALGORITHM_ATTR

This table defines the possible attributes for any algorithm.

*Table B–10 CIS_ALGORITHM_ATTR*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(100 | PK for this table. | N |
| NAME | VARCHAR2(50) | This is the name of the clustering algorithm attribute. There are some specific attributes per algorithm. (For example, BaNG uses attributes such as distance metric, converge factor, and number of iterations.) | N |
| KEYWORD_NAME | VARCHAR2(50) | This field contains the attribute/property name recognized internally by the application code. This value is used within the code to identify the property and must be provided with the correct case and format recognized within the code. | Y |
| DESCR | VARCHAR2(200) | Descriptive text for the algorithm. | Y |
| ALGORITHM_ATTR_TYPE_ID | NUMBER(10) | Algorithm attribute type ID. | N |
| DFLT_VALUE | VARCHAR2(80) | Default value to be used for the attribute. | Y |
| ATTR_SEQ | NUMBER(5) | Sequence in which the attribute should be processed/presented. | Y |
| CIS_ALGORITHM_ID | NUMBER(10) | This is the algorithm ID. This field is populated only for algorithm-specific properties/attributes; otherwise, this field is null. | Y |
| USER_PROVIDED_FLG | VARCHAR2(1) | A value of Y indicates that the user must set this value using the UI while creating the scenario configuration. | Y |
| CURRENT_FLG | VARCHAR2(1) | Indicates whether this row is the most recent row (Y), or if it is a historical row (N). | Y |
| DELETE_FLG | VARCHAR2(1) | Indicates whether the row is considered deleted (Y) or not (N). | Y |
| CREATED_BY_USER | VARCHAR2(255) | The user who created the record. | N |
| CREATED_ON_DT | TIMESTAMP(6) | The date when the record was created. | N |
| CHANGED_BY_USER | VARCHAR2(255) | The user who last changed the record. | Y |
| CHANGED_ON_DT | TIMESTAMP(6) | The date when the record was last changed. | Y |

**Notes:**

This table defines the possible attributes for any algorithm.

## CIS_ALGORITHM_ATTR_TYPE

Different types of algorithm properties/attributes are possible; some of them apply to multiple algorithms while other are algorithm specific.

*Table B–11 CIS_ALGORITHM_ATTR_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(50) | Algorithm attribute type name. | N |
| DESCR | VARCHAR2(200) | Algorithm attribute type description. | Y |

**Notes:**

Different types of algorithm properties/attributes are possible; some of them apply to multiple algorithms while others are algorithm specific.

Types include Common Properties, Scale Properties, Attribute Scale Properties, and Algorithm Specific Properties

## CIS_ATTR_TYPE

This table determines the possible attributes used in clustering, such as numeric or float.

*Table B–12 CIS_ATTR_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(50) | The name of attribute type. | N |
| DESCR | VARCHAR2(200) | Descriptive text for attribute type. | Y |
| GENERIC_TYPE_NAME | VARCHAR2(30) | Grouped attributes in generic name type. | Y |

**Notes:**

This table determines the possible attributes used in clustering, such as numeric or float.

## CIS_ATTR_TYPE_FN

This table determines the mathematical function that can be applied to the individual attributes based on their presence on different screens (pivot/insights).

*Table B–13 CIS_ATTR_TYPE_FN*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| SQL_FN | VARCHAR2(30) | The function applied on clustering attributes in different UI screens. | Y |
| SQL_DESCR | VARCHAR2(300) | Descriptive text to describe the clustering function. | Y |

**Notes:**

This table determines the mathematical function that can be applied on the individual attributes based on their presence on different screens (pivot/insights).

## CIS_BUS_OBJ_HIER_DEPLOY_XREF

This table identifies the business object, objectives, product hierarchy types, and levels that are eligible to be selected for deployment. This ensures that only authorized types of clusters are exported to external systems (Category Management).

*Table B–14 CIS_BUS_OBJ_HIER_DEPLOY_XREF*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BUSSOBJ_ID | NUMBER(10) | FK for the CIS_BUSINESS_OBJECT. Part of a composite FK to CIS_BUSSOBJ_OBJECTIVE_XREF. This defines the business object that is eligible for exporting. | N |
| OBJECTIVE_ID | NUMBER(10) | FK to the CIS_OBJECTIVE table. Part of a composite FK to the CIS_BUSSOBJ_OBJECTIVE_XREF table. This defines the objective that is eligible for being deployed. | N |
| PROD_HIER_TYPE_ID | NUMBER(10) | FK to RSE_HIER_TYPE, although it is expected to be limited to only hierarchy types that are related to products. | Y |
| PROD_HIER_LEVEL_ID | NUMBER(10) | FK to RSE_HIER_LEVEL. This defines the level of the product hierarchy within the specified product hierarchy type that is eligible for being deployed. | Y |
| CURRENT_FLG | VARCHAR2(1) | A Y/N flag to indicate whether this record is considered current (Y) or not (N). Only current records are able to be considered usable. | Y |
| DELETE_FLG | VARCHAR2(1) | A Y/N flag to indicate whether the record should be considered deleted (Y) or not (N). | Y |

## CIS_BUS_OBJ_NESTED_TCRITERIA

This table is used to determine possible child cluster type(s) for a parent cluster.

*Table B–15 CIS_BUS_OBJ_NESTED_TCRITERIA*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BUSSOBJ_ID | NUMBER(10) | FK for the business object objective. | N |
| OBJECTIVE_ID | NUMBER(10) | FK for the objective. | N |
| TYPE_CRITERIA_ID | NUMBER(10) | FK for the type criteria. | N |
| CHL_TYPE_CRITERIA_ID | NUMBER(10) | The possible child criteria ID. It is mainly used for nested clustering. | N |
| CHL_CRITERIA_ALLWD_FLG | VARCHAR2(1) | Indicates whether the combination of parent and child is allowed in a given matrix. | Y |
| CURRENT_FLG | VARCHAR2(1) | Indicate whether this row is the most recent row (Y), or if it is a historical row (N). | Y |
| DELETE_FLG | VARCHAR2(1) | Indicates whether the row is considered deleted (Y) or not (N). | Y |

**Notes:**

This table is used to determine possible child cluster type(s) for a parent cluster. It is mainly used in nested clustering.

## CIS_BUS_OBJ_TCRITERIA_ATT_XREF

This table is a cross reference for the possible attributes based on objective, business objective, cluster type, and product.

*Table B–16 CIS_BUS_OBJ_TCRITERIA_ATT_XREF*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| BUSSOBJ_ID | NUMBER(10) | FK for the business object objective. | N |
| OBJECTIVE_ID | NUMBER(10) | FK for the CIS objective. | N |
| TYPE_CRITERIA_ID | NUMBER(10) | Cluster types allowed for the combination of objective ID and business objective ID. | N |
| TCRITERIA_ATTR_ID | NUMBER(10) | This field contain the attribute ID reference from cis_tcriteria_attr table. | N |
| DFLT_ATTR_WEIGHT | NUMBER(18,4) | Default weight to be used for an attribute. | Y |
| PARTICIPATING_ATTR_FLG | VARCHAR2(1) | Identifies those attributes within the type that are considered as optimal for the clustering calculation. | Y |
| VALIDATING_ATTR_FLG | VARCHAR2(1) | This flag is an indicator for the type of attribute. This indicates that it is a validating attribute. | Y |
| INFORMATIONAL_ATTR_FLG | VARCHAR2(1) | Identifies those attributes within the type that are considered as only informational but not optimal for the clustering calculation. | Y |
| ATTR_SEQ | NUMBER(4) | Internal use for ordering of attributes. | Y |
| CURRENT_FLG | VARCHAR2(1) | Indicates whether this row is the most recent row (Y), or if it is a historical row (N). | Y |
| DELETE_FLG | VARCHAR2(1) | Indicates whether the row is considered deleted (Y) or not (N). | Y |
| CREATED_BY_USER | VARCHAR2(255) | The user who created the record. | N |
| CREATED_ON_DT | TIMESTAMP(6) | The date when the record was created. | N |
| CHANGED_BY_USER | VARCHAR2(255) | The user who last changed the record. | Y |
| CHANGED_ON_DT | TIMESTAMP(6) | The date when the record was last changed. | Y |
| PROD_HIER_ID | NUMBER(10) | This field is populated only for product attribute clustering, as the attributes do not change for other cluster types. | Y |
| PROD_HIER_TYPE_ID | NUMBER(10) | This field is populated only for product attribute clustering, as the attributes do not change for other cluster types. | Y |
| FN_STORE_LVL | NUMBER(10) | This field determines the aggregation level at store (Default=2). | Y |
| FN_CLUSTER_LVL | NUMBER(10) | This field determines the aggregation level at cluster (Default=2). | Y |
| FN_PIVOT_STORE_LVL | NUMBER(10) | This field determines the aggregation level at explore data screen (Default=2). | Y |
| TCRITERIA_GRP_BY_ID | NUMBER(10) | This column is used to for grouping of certain type criteria(s) for Mixed Attribute clustering. | Y |

*Table B–16 (Cont.) CIS_BUS_OBJ_TCRITERIA_ATT_XREF*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ATTR_DATE_VALUE | DATE | This field contains the provided date value for product attribute group. | Y |
| ATTR_NUM_VALUE | NUMBER(22,5) | This field contains the provided number value for product attribute group. | Y |
| ATTR_STRING_VALUE | VARCHAR2(255) | This field contains the provided string value for product attribute group. | Y |
| ATTR_VALUE_EXT_CODE | VARCHAR2(255) | This field contains the externally provided unique ID for this product attribute group. | Y |
| PROD_ATTR_GRP_ID | NUMBER(10) | Referenced product attribute ID for product attribute. | Y |
| PROD_ATTR_GRP_NAME | VARCHAR2(255) | Referenced product attribute group name for product attribute. | Y |
| PROD_ATTR_GRP_VALUE_ID | NUMBER(10) | Referenced product attribute value ID for product attribute. | Y |

**Notes:**

This table is a cross reference for the possible attributes based on objective, business objective, cluster type and product.

It also includes the information about whether the attribute is informational or participating.

## CIS_BUS_OBJ_TCRITERIA_XREF

This table specifies the possible cluster types allowed for the combination of objective ID and business objective ID.

*Table B–17 CIS_BUS_OBJ_TCRITERIA_XREF*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BUSSOBJ_ID | NUMBER(10) | FK for the business object objective. | N |
| OBJECTIVE_ID | NUMBER(10) | FK for the CIS objective. | N |
| TYPE_CRITERIA_ID | NUMBER(10) | Type Criteria ID. | N |
| PRIMARY_FLG | VARCHAR2(1) | This is the primary/default cluster objective type to be used. | Y |
| BUS_OBJ_TCRITERIA_SEQ | NUMBER(5) | This is the sequence in which the instance should be used in cases where there are multiple instances. | Y |
| DEF_SCRE_THRESHOLD_PCT | NUMBER(7,4) | This is a default threshold for the score for each criteria, based on which the UI indicates if a score is below a desired threshold. If not provided, the default score threshold from RSE_CONFIG is used, which is applicable across all the criteria. | Y |
| OUTLIER_RULE_ID | NUMBER(10) | FK to the CIS outlier rules. | N |
| OUTLIER_ATTR_VALUE_1 | NUMBER(7,4) | The value that defines the threshold for which a data value is considered an outlier for the criteria. If not specified, then the outlier value will be obtained from RSE_CONFIG. | Y |

*Table B–17 (Cont.) CIS_BUS_OBJ_TCRITERIA_XREF*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| OUTLIER_ATTR_VALUE_2 | NUMBER(7,4) | An additional value that can be used to define the threshold for which a data value is considered an outlier for the criteria. If not specified, then the outlier value will be obtained from RSE_CONFIG. | Y |
| OUTLIER_ATTR_VALUE-3 | NUMBER(7,4) | An additional value that can be used to define the threshold for which a data value is considered an outlier for the criteria. If not specified, then the outlier value will be obtained from RSE_CONFIG. | Y |
| NEWSTR_RULE_ID | NUMBER(10) | FK to the CIS_NEWSTR_RULES. | N |
| DISABLE_ATTR_WTS | VARCHAR2(1) | A Y/N flag to enable (N) or disable (Y) attribute weights. | Y |
| INSIGHT_LINE_GRAPH_BI | VARCHAR2(1) | A Y/N flag to enable (N) or disable (Y) display line graph in insight screen. | Y |
| INSIGHT_SLS_CENTROID_GRAPH_BI | VARCHAR2(1) | A Y/N flag to enable (N) or disable (Y) sales centroid graph in insight screen. | Y |
| INSIGHT_VALIDATION_GRAPH_BI | VARCHAR2(1) | A Y/N flag to enable (N) or disable (Y) validation graph in insight screen. | Y |
| INSIGHT_SALES_GRAPH_BI | VARCHAR2(1) | A Y/N flag to enable (N) or disable (Y) sales graph in insight screen. | Y |
| INSIGHT_STACK_GRAPH_BI | VARCHAR2(1) | A Y/N flag to enable (N) or disable (Y) stack bar graph in insight screen. | Y |
| INSIGHT_SHOW_LG_BI | VARCHAR2(1) | A Y/N flag to display (Y) or not display (N) display line graph in insight screen. | Y |
| INSIGHT_SHOW_SCG_BI | VARCHAR2(1) | A Y/N flag to display (Y) or not display (N) sales centroid graph in insight screen. | Y |
| INSIGHT_SHOW_VG_BI | VARCHAR2(1) | A Y/N flag to display (Y) or not display (N) validation graph in insight screen. | Y |
| INSIGHT_SHOW_SG_BI | VARCHAR2(1) | A Y/N flag to display (Y) or not display (N) sales graph in insight screen. | Y |
| INSIGHT_SHOW_SB_BI | VARCHAR2(1) | A Y/N flag to display (Y) or not display (N) stack bar graph in insight screen. | Y |
| SRC_TIME_PERIOD_FLG | VARCHAR2(1) | A Y/N flag to show display (Y) or not display (N) time period in definition screen. | Y |
| SUMM_FLG | VARCHAR2(1) | A Y/N flag to show display (Y) or not display (N) calendar summarization in definition screen. | Y |
| MERCH_GRP_FLG | VARCHAR2(1) | A Y/N flag to show display (Y) or not display (N) product summarization in definition screen. | Y |
| CURRENT_FLG | VARCHAR2(1) | A flag to indicate whether this row is the most recent row (Y), or if it a historical row (N). | Y |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether the row is considered deleted (Y) or not (N). | Y |
| CREATED_BY_USER | VARCHAR2(255) | User who created the record. | N |
| CREATED_ON_DT | TIMESTAMP(6) | Record creation date. | N |

*Table B–17 (Cont.) CIS_BUS_OBJ_TCRITERIA_XREF*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| CHANGED_BY_USER | VARCHAR2(255) | User who last changed the record. | Y |
| CHANGED_ON_DT | TIMESTAMP(6) | Last changed date. | Y |
| PIVOT_TBL_FLTR_FLG | VARCHAR2(1) | A Y/N to show display (Y) or not display (N) filters in UI | Y |

**Notes:**

This table specifies the possible cluster types allowed for the combination of objective ID and business objective ID.

## CIS_BUSINESS_OBJECT

This table hosts the list of application that are registered/configured to use the clustering feature. Values are provided for Life Cycle Pricing and Promotion (LCPP), Category Management (CM), and Assortment Rationalization (AR).

*Table B–18 CIS_BUSINESS_OBJECT*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | |
| NAME | VARCHAR2(50) | This is the name of the business objects that is used to generate the clusters (for example, Store Cluster, Customer Cluster). | |
| SHORT_NAME | VARCHAR2(10) | Short name of the business objective. | |
| DESCR | VARCHAR2(200) | Description of the business objective. | |
| SOURCE_ENTITY_NAME | VARCHAR2(30) | The name of the database table that provides the values for this business object. | |
| SOURCE_COLUMN_NAME | VARCHAR2(30) | The name of the database table column that provides the values for this business object. | |
| SOURCE_ENTITY_ALIAS | VARCHAR2(30) | An alias/abbreviation by which this source database object should be referred t0. | |
| CURRENT_FLG | VARCHAR2(1) | A flag to indicate whether this row is the most recent row (Y), or if it a historical row (N). | |
| CREATED_BY_USER | VARCHAR2(255) | User who created the record. | |
| CREATED_ON_DT | TIMESTAMP(6) | Record creation date. | |
| CHANGED_VY_USER | VARCHAR2(255) | User who last changed the record. | |
| CHANGED_ON_DT | TIMESTAMP(6) | Last changed date. | |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether the row is considered deleted (Y) or not (N). | |

**Notes:**

This table hosts the list of application that are registered/configured to use the clustering feature. Values include Life Cycle Pricing and Promotion (LCPP), Category Management (CM), and Assortment Rationalization (AR).

## CIS_BUSSOBJ_OBJ_ALG_XREF

This cross reference table provides the ability to use the same algorithm to generate different cluster objectives. The same algorithm can be used to generate customer clusters as well as store clusters. In addition, multiple algorithms can be listed that can be used to achieve a similar objective.

***Table B–19 CIS_BUSSOBJ_OBJ_ALG_XREF***

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BUSSOBJ_ID | NUMBER(10) | FK to the business object objective. | N |
| OBJECTIVE_ID | NUMBER(10) | FK for the CIS Objective | N |
| ALGORITHM_ID | NUMBER(10) | FK for the Algorithm to be applied to business ID and objective ID. | N |
| PRIMARY_FLG | VARCHAR2(1) | This flag identified the primary attribute within the type. | Y |
| BUSSOBJ_OBJ_ALG_SEQ | NUMBER(4) | This field provides the sequence in which the specific cross reference should be used. | Y |
| CURRENT_FLG | VARCHAR2(1) | A flag to indicate whether this row is the most recent row (Y), or if it a historical row (N). | Y |
| CREATED_BY_USER | VARCHAR2(255) | User who created the record. | N |
| CREATED_ON_DT | TIMESTAMP(6) | Record creation date. | N |
| CHANGED_BY_USER | VARCHAR2(255) | User who last changed the record. | Y |
| CHANGED_ON_DT | TIMESTAMP(6) | Last changed date. | Y |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether the row is considered deleted (Y) or not (N). | Y |

**Notes:**

This table provide the ability to 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, multiple algorithms that could be used to achieve similar objective can be listed.

ALGORITHM_ID - Algorithm to be applied to BUSINESS_ID and OBJECTIVE_ID.

## CIS_BUSSOBJ_OBJECTIVE_XREF

This table hosts the cross reference for the clustering objectives that are valid for an application. In most cases, an application will use clustering feature to achieve a single objective; however, there could be cases in which an application can perform cluster generation for multiple purposes.

***Table B–20 CIS_BUSSOBJ_OBJECTIVE_XREF***

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BUSSOBJ_ID | NUMBER(10) | BUSSOBJ_ID is reference to CIS_BUSINESS_OBJECT table. | N |
| OBJECTIVE_ID | NUMBER(10) | OBJECTIVE_ID refers to CIS_OBJECTIVE table. | N |
| PRIMARY_FLG | VARCHAR2(1) | This is the primary/default cluster objective to be used. | Y |

*Table B–20 (Cont.) CIS_BUSSOBJ_OBJECTIVE_XREF*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BUSSOBJ_OBJ_SEQ | NUMBER(5) | This is the sequence at which the instance should be used in case there are multiple instances. | Y |
| CURRENT_FLG | VARCHAR2(1) | A flag to indicate whether this row is the most recent row (Y), or if it a historical row (N). | Y |
| CREATED_BY_USER | VARCHAR2(255) | User who created the record. | N |
| CREATED_ON_DT | TIMESTAMP(6) | Record creation date. | N |
| CHANGED_BY_USER | VARCHAR2(255) | User who last changed the record. | Y |
| CHANGED_ON_DT | TIMESTAMP(6) | Last changed date. | Y |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether the row is considered deleted (Y) or not (N). | Y |

**Notes:**

This table contains cross references about which clustering objectives are valid per application.

BUSSOBJ_ID refers to CIS_BUSINESS_OBJECT table.

OBJECTIVE_ID refers to CIS_OBJECTIVE table.

BUSSOBJ_OBJ_SEQ determines the sequence in which the instance should be used in case there are multiple instances.

## CIS_BUSSOBJ_TCRIT_HIER_XREF

This table specifies the possible hierarchy levels for each hierarchy type (merchandise and location) allowed for the combination of objective ID, business objective ID, and type criteria ID.

*Table B–21 CIS_BUSSOBJ_TCRIT_HIER_XREF*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BUSSOBJ_ID | NUMBER(10) | FK for the business object objective. | N |
| OBJECTIVE_ID | NUMBER(10) | FK for the CIS objective. | N |
| TYPE_CRITERIA_ID | NUMBER(10) | Type Criteria ID. | N |
| HIER_TYPE_ID | NUMBER(10) | FK for the RSE_HIER_LEVEL table hier_type_id. | N |
| HIER_LVL_ID | NUMBER(4) | FK for the RSE_HIER_LEVEL table ID. | N |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether the row is considered deleted (Y) or not (N). | Y |

## CIS_CLUSTER_CATEGORY

This table determines the possible type of cluster types - normal, flagship, dormant etc

*Table B–22 CIS_CLUSTER_CATEGORY*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(80) | Name of the cluster category. | Y |
| DESCR | VARCHAR2(255) | Special Cluster - Manually Allocated, Special Cluster - Exceptional, Normal Cluster. | Y |

**Notes:**

This table determines the possible type of cluster types, such as normal, flagship, or dormant.

## CIS_CLUSTER_GROUP_BY_TYPE

This table contains the catalog for the distinct group-by options available for clustering.

*Table B–23 CIS_CLUSTER_GROUP_BY_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(50) | The name of the group by option. | Y |
| DESCR | VARCHAR2(80) | This entity contains the catalog for the distinct group by options available for clustering, including Group By, Location Node, Merchandise Node, Calendar Node, Location Hierarchy Level, Merchandise Hierarchy Level, Calendar Hierarchy Level. | Y |

**Notes:**

This table keeps the catalog for the distinct group by options available for clustering, including Group By, Location Node, Merchandise Node, Calendar Node, Location Hierarchy Level, Merchandise Hierarchy Level, and Calendar Hierarchy Level.

## CIS_CLUSTER_MEMBER_INT

This table contains the final stores created by the clusters approved for a given combination of location, merchandise, calendar dimension(s), and cluster criteria type.

*Table B–24 CIS_CLUSTER_MEMBER_INT*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| CLUSTER_ID | NUMBER(10) | This column is populated by cis_cluster.id. | N |
| CLUSTER_LABEL | VARCHAR2(50) | This column is populated by cis_cluster.name. | N |
| CLUSTER_SET_ID | NUMBER(10) | This column is populated by cis_cluster_config_hdr.id. | N |
| STORE_LOCATION_ID | VARCHAR2(80) | This column is populated by rse_loc_src_xref.loc_ext_key. | N |

**Notes:**

This table contains the final stores created by the clusters approved for a given combination of location, merchandise, calendar dimension(s), and cluster criteria type.

## CIS_CLUSTER_OUTLIER_RULE

This table contains the possible outlier rules for a type criteria.

*Table B–25 CIS_CLUSTER_OUTLIER_RULE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| BUSSOBJ_ID | NUMBER(10) | FK for the business object objective. | N |
| TYPE_CRITERIA_ID | NUMBER(10) | FK for the type criteria. | N |
| OBJECTIVE_ID | NUMBER(10) | FK for the CIS objective. | N |
| DESCR | VARCHAR2(200) | Descriptive text to define this outlier rule. | Y |
| RULE | VARCHAR2(80) | This is the text to use to evaluate the rule store_count > X.<br>X is a placeholder that is replaced by the value provided by the user or the default value set during initial installation. | Y |
| DFLT_LOW_VALUE | VARCHAR2(50) | This defines the minimum value for this outlier rule. For example, this can specify that a minimum of five locations are required in a cluster. | Y |
| DFLT_HIGH_VALUE | VARCHAR2(50) | This defines the maximum value for this outlier rule. For example, this can specify that a maximum of 50 locations are allowed in a cluster. | Y |
| ACTIVE_FLG | VARCHAR2(1) | This is a Y/N flag that defines whether the rule is an actively used rule (Y) or not (N). | Y |
| CURRENT_FLG | VARCHAR2(1) | A flag to indicate whether this row is the most recent row (Y), or if it a historical row (N). | Y |
| DELETE_FLG | VARCHAR2(1) | A Y/N flag to indicate whether this row is considered a deleted row (Y) or not (N). | Y |
| CREATED_BY_USER | VARCHAR2(255) | User who created the record. | N |
| CREATED_ON_DT | TIMESTAMP(6) | Record creation date. | N |
| CHANGED_BY_USER | VARCHAR2(200) | User who last changed the record. | Y |
| CHANGED_ON_DT | TIMESTAMP(6) | Last changed date. | Y |

**Notes:**

This table contains the possible outlier rules for a type criteria.

## CIS_CLUSTER_SET_EXP_VW

This view provides an exportable set of clusters to send to Category Management.

*Table B–26 CIS_CLUSTER_SET_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| EFF_START_DT | VARCHAR2(11) | The starting date when the cluster is effective. | Y |
| EFF_END_DT | VARCHAR2(11) | The ending date when the cluster is effective. | Y |
| PROD_EXT_KEY | VARCHAR2(80) | The external ID for the product hierarchy this cluster is applicable to. | N |
| LOC_EXT_KEY | VARCHAR2(80) | The external ID for the store location that belongs in this cluster. | N |
| CLUSTER_ID | NUMBER(10) | The identifier for the cluster. | N |
| CLUSTER_LABEL | VARCHAR2(50) | A descriptive name/label for the cluster. | N |

## CIS_CLUSTER_SET_INT

This table contains the approved clusters for a given combination of location, merchandise, calendar dimension(s), and cluster criteria type.

*Table B–27 CIS_CLUSTER_SET_INT*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| CLUSTER_SET_ID | NUMBER(10) | This column is populated by cis_cluster_config_hdr.id. | N |
| CLUSTER_SET_LABEL | VARCHAR2(50) | This column is populated by cis_cluster_config_hdr.name. | N |
| PARENT_CLUSTER_SET_ID | NUMBER(10) | This column is populated by cis_cluster_config_hdr.parent_cluster_config_hdr_id. | N |
| MERCHANDISE_ID | VARCHAR2(80) | Thsi column is populated by rse_prod_src_xref.prod_ext_key. | N |
| LOCATION_ID | VARCHAR2(80) | This column is populated by rse_loc_src_xref.loc_ext_key. | N |
| EFFECTIVE_START_DATE | DATE | Start date of the given combination of clusters. | N |
| EFFECTIVE_END_DATE | DATE | End date of the given combination of clusters. | N |

**Notes:**

This table contains the approved clusters for a given combination of location, merchandise, calendar dimension(s), and cluster criteria type.

## CIS_CLUSTER_STATUS

This table determines the status of the cluster.

*Table B–28 CIS_CLUSTER_STATUS*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for the table. | N |
| DESCR | VARCHAR2(80) | Descriptive text for this cluster status. | Y |

**Notes:**

This table determines the status of the cluster.

## CIS_CLUSTER_TYPE

This table contains the list of the different types of possible clusters, including deployed, automatically generated, and user generated.

*Table B–29 CIS_CLUSTER_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| DESCR | VARCHAR2(80) | Descriptive text for this cluster type. | Y |

**Notes:**

This table contains the list of the different types of possible clusters, including deployed, automatically generated, and user generated.

## CIS_EFFECTIVE_PERIOD

This table contains the planning period information.

*Table B–30 CIS_EFFECTIVE_PERIOD*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| DESCR | VARCHAR2(80) | Descriptive text for this effective period. | Y |
| START_PERIOD_ID | NUMBER(10) | Start period ID in the calendar hierarchy for the period. | N |
| END_PERIOD_ID | NUMBER(10) | End period ID in the calendar hierarchy for the period. | N |
| BUSSOBJ_ID | NUMBER(10) | FK for the business object objective. | N |
| OBJECTIVE_ID | NUMBER(10) | FK for the objective . | N |
| TYPE_CRITERIA_ID | NUMBER(10) | FK for the type criteria. | N |
| ACTIVE_FLG | VARCHAR2(1) | This is a Y/N flag that defines whether this period is to be actively used (Y) or not (N). | Y |

**Notes:**

This table contains the planning period information.

## CIS_OBJECTIVE

This table contains the details corresponding to the different objectives that can be achieved by the clustering implementation.

*Table B–31 CIS_OBJECTIVE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(50) | This is the purpose/usage for which the cluster is created. | N |
| DESCR | VARCHAR2(200) | More descriptive text for this objective. | Y |
| CURRENT_FLG | VARCHAR2(1) | A flag to indicate whether this row is the most recent row (Y), or if it a historical row (N). | Y |
| CREATED_BY_USER | VARCHAR2(255) | User who created the record. | N |
| CREATED_ON_DT | TIMRESTAMP(6) | Record creation date. | N |
| CHANGED_BY_USER | VARCHAR2(255) | User who last changed the record. | Y |
| CHANGED_ON_DT | TIMESTAMP(6) | Last changed date. | Y |
| DELETE_FLG | VARCHAR2(1) | A Y/N flag to indicate whether this record is considered deleted (Y) or not (N). | Y |

**Notes:**

This table contains details corresponding to the different objectives that can be achieved by the clustering implementation.

## CIS_SCENARIO_STATUS

This table contains the scenario status, including created, executed, approved or rejected.

*Table B–32 CIS_SCENARIO_STATUS*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| DESCR | VARCHAR2(80) | Descriptive text for the scenario. | Y |

**Notes:**

This table contains the scenario status.

## CIS_TCRITERIA_SRC_XREF

This table is a cross reference for SRC_ENTITY_NAME and also the settings for Participating, Informational for that SRC_ENTITY_NAME's attributes.

*Table B–33 CIS_TCRITERIA_SRC_XREF*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BUSSOBJ_ID | NUMBER(10) | PK for this table. | N |
| OBJECTIVE_ID | NUMBER(10) | PK for this table. | N |
| TYPE_CRITERIA_ID | NUMBER(10) | PK for this table. | N |
| SRC_ENTITY_NAME | VARCHAR2(30) | The name of the DB view which should be including when setting up the BOTAX table data. PK for this table | N |

*Table B–33 (Cont.) CIS_TCRITERIA_SRC_XREF*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PARTICIPATING_ATTR_FLG | VARCHAR2(1) | A Y/N flag to indicate whether the attributes from this DB view should be participating (Y) or not (N). Only one ATTR_FLG column can be set to a Y value. | N |
| VALIDATING_ATTR_FLG | VARCHAR2(1) | A Y/N flag to indicate whether the attributes from this DB view should be validating (Y) or not (N). | N |
| INFORMATIONAL_ATTR_FLG | VARCHAR2(1) | A Y/N flag to indicate whether the attributes from this DB view should be informational (Y) or not (N). Only one ATTR_FLG column can be set to a Y value. | N |
| CURRENT_FLG | VARCHAR2(1) | A Y/N flag to indicate whether this row should be considered as a currently usable row (Y) or not (N). | Y |
| DELETE_FLG | VARCHAR2(1) | A Y/N flag to indicate whether this row is considered deleted (Y) or not (N). | Y |

**Notes:**

This table contains the scenario status.

## CIS_TCRITERIA_ATTR

This table contains all the possible attributes in clustering process.

*Table B–34 CIS_TCRITERIA_ATTR*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(50) | Name for the criteria attribute. | N |
| DESCR | VARCHAR2(200) | More descriptive text for the criteria attribute. | Y |
| SRC_ENTITY_NAME | VARCHAR2(30) | Name of table/view that holds the attribute information/value coming from Retail Analytics or an external system. | Y |
| SRC_COLUMN_NAME | VARCHAR2(30) | Name of the column in the Retail Analytics or external system table/view. | Y |
| SRC_ENTITY_ALIAS | VARCHAR2(30) | Alias of the column in the Retail Analytics or external system table/view. | Y |
| ATTR_TYPE_ID | NUMBER(10) | Type of the attribute. | N |
| DISPLAY_PERCENT | VARCHAR2(1) | Display percent symbol in BI for an attribute. | Y |
| DISPLAY_FORMAT_ID | NUMBER(10) | Foreign key to the RSE_DISPLAY_FORMAT table that enables the control of how the attribute should be formatted for display.<br>NOTE: The control file expects a string that corresponds to the EXT_KEY column of the RSE_DISPLAY_FORMAT table. | Y |
| CURRENT_FLG | VARCHAR2(1) | A flag to indicate whether this row is the most recent row (Y), or if it a historical row (N). | Y |

*Table B–34 (Cont.) CIS_TCRITERIA_ATTR*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether the row is considered deleted (Y) or not (N). | Y |
| CREATED_BY_USER | VARCHAR2(255) | User who created the record. | N |
| CREATED_ON_DT | TIMESTAMP(6) | Record creation date. | N |
| CHANGED_BY_USER | VARCHAR2(255) | User who last changed the record. | Y |
| CHANGED_ON_DT | TIMESTAMP(6) | Last changed date. | Y |
| UI_DISPLAY_NAME | VARCHAR2(200) | The value to display in the UI when showing this criteria attribute. | Y |
| TCRITERIA_ATTR_SRC_XREF_ID | NUMBER(10) | Field contains the reference to the PK of the of the object which this attribute definition belongs to. For example, if an attribute is representative of a specific consumer segment, then this column will contain the FK to that consumer segment. | Y |

**Notes:**

This table contains all the possible attributes in all type of clustering process.

## CIS_TCRITERIA_ATTR_TYPE_VALUE

This table is used to maintain the list of valid values for ordinal and categorical attributes. The list of valid values is directly assigned to the type criteria attribute, since the valid values are different for each attribute.

*Table B–35 CIS_TCRITERIA_ATTR_TYPE_VALUE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| TCRITERIA_ATTR_ID | NUMBER(10) | FK of the cis_tcriteria_attr entity. | N |
| ATTR_VALUE | VARCHAR2(80) | This column contains the attribute value. | Y |
| VALUE_SEQ | NUMBER(4) | This column contains the sequence in which the values should be used/displayed. | Y |
| VALUE_PRIORITY_RANK | NUMBER(4) | This column is used as a tiebreaker in cases where two values have the same number of instances and the screen.process needs to break the tie by giving preference to one value over the other. | Y |

**Notes:**

This table is used to maintain the list of valid values for ordinal and categorical attributes. The list of valid values is directly assigned to the type criteria attribute, since the valid values is different for each attribute.

## CIS_TCRITERIA_ATTR_XREF

This table contains the cross reference between attributes for the cluster types.

*Table B–36 CIS_TCRITERIA_ATTR_XREF*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| TCRITERIA_ATTR_ID | NUMBER(10) | This field contains the possible attribute defined in clustering. | N |
| TYPE_CRITERIA_ID | NUMBER(10) | This field contains the cluster type defined against each clustering attribute. | N |
| CURRENT_FLG | VARCHAR2(1) | A flag to indicate whether this row is the most recent row (Y), or if it a historical row (N). | Y |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether the row is considered deleted (Y) or not (N). | Y |
| CREATED_BY_USER | VARCHAR2(255) | User who created the record. | N |
| CREATED_ON_DT | TIMESTAMP(6) | Record creation date. | N |
| CHANGED_BY_USER | VARCHAR2(255) | User who last changed the record. | Y |
| CHANGED_ON_DT | TIMESTAMP(6) | Last changed date. | Y |

**Notes:**

This table contains the cross reference between attributes for the cluster types.

## CIS_TYPE_CRITERIA

This table contains the list of different clustering types/criteria that can be used to generate clusters.

*Table B–37 CIS_TYPE_CRITERIA*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(50) | This is the UI name of the clustering type/criteria. This is the cluster by list of values. | N |
| TYPE_NAME | VARCHAR2(50) | This is the static name of the clustering type/criteria. This should never be changed for display purposes. | N |
| DESCR | VARCHAR2(200) | More descriptive text to describe the criteria type. | Y |
| CODE | CHAR | Internal identification code for this clustering type/criteria. | N |
| DISTRIBUTION_GRP_FLG | VARCHAR2(1) | Certain attribute can be represented as a distribution (per store and/or cluster, such as with consumer segmentation). This flag helps the UI identify which attributes belong to the same distribution group (where the total value adds up to 100%). | Y |
| DFLT_MIN_NUM_CLUSTER | NUMBER(10) | This is the default for the minimum number of clusters expected for this cluster type. | Y |
| DFLT_MAX_NUM_CLUSTER | NUMBER(10) | This is the default for the maximum number of clusters expected for this cluster type. | Y |
| CURRENT_FLG | VARCHAR2(1) | A flag to indicate whether this row is the most recent row (Y), or if it a historical row (N). | Y |

*Table B–37 (Cont.) CIS_TYPE_CRITERIA*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| CREATED_BY_USER | VARCHAR2(255) | User who created the record. | N |
| CREATED_ON_DT | TIMESTAMP(6) | Record creation date. | N |
| CHANGED_BY_USER | VARCHAR2(255) | User who last changed the record. | Y |
| CHANGED_ON_DT | TIMESTAMP(6) | Last changed date. | Y |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether the row is considered deleted (Y) or not (N). | Y |

**Notes:**

This table contains the list of different clustering types/criteria that can be used to generate clusters.

## DT_AE_EXP_VW

This view provides all the exportable data elements needed to provide the Assortment Elasticity metric that the DT application has calculated. This view only provides results for active results.

*Table B–38 DT_AE_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROD_EXT_KEY | VARCHAR2(80) | The external identifier for the category this data belongs to. | Y |
| CUSTSEG_EXT_KEY | VARCHAR2(80) | The external identifier for the customer segment associated with this data. | Y |
| LOC_EXT_KEY | VARCHAR2(80) | The external identifier for the location | Y |
| AE | NUMBER(22,7) | The assortment elasticity that DT has calculated. | Y |
| EFFECTIVE_DT_FROM | DATE | The date when this data was activated. | Y |
| EFFECTIVE_DT_TO | DATE | Not used. | Y |

## DT_ASSORT_MULT_EXP_VW

This view provides an exportable list of assortment multipliers to RDF so that the impact of assortment changes can influence RDF results.

*Table B–39 DT_ASSORT_MULT_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| EFF_START_DT | DATE | The date that this assortment multiplier is effective for use. | Y |
| PROD_EXT_KEY | VARCHAR2(80) | The external ID for the product. | Y |
| LOC_EXT_KEY | VARCHAR2(80) | The external ID for the store location. | Y |
| ASSORT_MULT | NUMBER(38,20) | The assortment multiplier associated with the product and location, as a result of changes to the assortment at the location. | Y |

## DT_ATTR_WGT_EXP_VW

This view provides the export data that is provided to the Category Management system for attribute weights used by the DT application.

*Table B–40 DT_ATTR_WGT_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROD_EXT_KEY | VARCHAR2(80) | The external ID for the category | Y |
| CUSTSEG_EXT_KEY | VARCHAR2(80) | The external ID for the customer segment | Y |
| LOC_EXT_KEY | VARCHAR2(80) | The external ID for the location hierarchy/trade area | Y |
| ATTRIBUTE_EXTERNAL_ID | VARCHAR2(80) | The external ID for the attribute | N |
| ATTR_WGT | NUMBER(22,7) | The decimal weight that this attribute represents within the category/customer segment/trade area. | Y |
| FUNC_ATTR_IND | NUMBER | An indicator to indicate if the attribute is a functional attribute (1) or not (0). A functional attribute is one that fits a specific purpose and cannot be substituted by other products with other values for this attribute. | Y |

## DT_EXCLUDE

This table defines the different types of pruning filters that are available to exclude a DT result from being used during the escalation phase of the DT workflow.

*Table B–41 DT_EXCLUDE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(80) | The name of the pruning filter. | Y |
| DESCR | VARCHAR2(255) | Descriptive text for the pruning filter. | Y |
| PROC_NAME | VARCHAR2(30) | Name of the process that performs this data filter.ing | Y |
| DISPLAY_SEQ | NUMBER(10) | The order in which the filters should be displayed when viewing filter statistics in the UI. | Y |
| PROCESS_SEQ | NUMBER(10) | The order in which the filters should be executed during the filter stage. | Y |
| ENABLE_FLG | VARCHAR2(1) | A Y/N flag that indicates whether the filter is enabled for use. | Y |

**Notes:**

This table can be used to customize the pruning filters that run after DT Models have been created. This allows for the flexibility in the customized rules for pruning DT Models from the list of usable Models.

The NAME and DESCR columns are displayed in the UI if the NAME value is not found in the UI's resource bundle.

The PROC_NAME is the name of the procedure that should be executed to perform this pruning task. This routine should be implemented in either the DT_PRUNE_T type, or in a database type that extends this type and then is configured as the type to execute for the DT_PRUNE database service.

When the pruning stage runs, this table is referenced, and each enabled filter is executed in ascending order based on the PROCESS_SEQ column. Therefore, this column should be unique in order to ensure consistency in the order in which the filters are executed.

The DISPLAY_SEQ column allows for displaying the filters in the UI in a different order than what they may be executed. Although setting these two columns with different values might be confusing to the user, it can be done if needed.

The ENABLE_FLG offers the ability to exclude a filter from processing. This can be helpful if a filter is no longer deemed relevant to run. Setting this column to N prevents further execution of the filter, although any existing filter statistics about prior executions of the filter will remain.

## DT_FILTER

This table defines the different types of data filters that are used during the DT data filtering process.

*Table B–42 DT_FILTER*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | The PK for the DT Filter. | N |
| NAME | VARCHAR2(80) | A name for the filter. | Y |
| DESCR | VARCHAR2(255) | Descriptive text to describe the filter. | Y |
| SRVC_TYPE | VARCHAR2(30) | The service type for the data filters implementation. | Y |
| SRVC_NAME | VARCHAR2(30) | The service name for the data filters implementation. | Y |
| DISPLAY_SEQ | NUMBER(10) | The order in which the filters should be displayed in the UI. | Y |
| PROCESS_SEQ | NUMBER(10) | The order in which the filters should be executed. | Y |
| ENABLE_FLG | VARCHAR2(1) | A Y/N flag to indicate whether the filter is enabled (Y) or not (N) when data filtering is executed. | Y |

**Notes:**

This table contains the list of DT Filters that can be executed against baseline sales data in order to remove outliers or any data that can lead to bad results. Although the UI has a fixed number of prompts on the data filtering stage, this table offers the ability to add or remove any filter in the data filtering workflow.

When the data filtering stage runs, this table will be referenced and each enabled filter will be executed in ascending order, based on the PROCESS_SEQ column. Therefore, this column should be unique in order to ensure consistency in the order in which the filters are executed.

The DISPLAY_SEQ column allows for displaying the filters in the UI in a different order than the one they may be executed in. Although setting these two columns with different values might be confusing to the user, it can be done if needed.

The ENABLE_FLG offers the ability to exclude a filter from processing. This can be helpful if a filter is no longer deemed relevant to run. Setting this column to N prevents further execution of the filter, although any existing filter statistics about prior executions of the filter will remain.

The SRVC_TYPE and SRVC_NAME columns allow for the specification of the actual implementation that performs the data filtering task. These implementations should be extensions of the DT_FILTER_BASELINE_T database type. The SRVC_TYPE should be defined in the RSE_SRVC_TYPE table, and the SRVC_NAME should be defined in the RSE_SRVC_CONFIG table. Because of this configuration, this table offers the ability to run additional baseline sales filters.

## DT_LOC_WK_EXCL_STG

This is a staging table that loads a list of locations and dates that should be excluded from Demand Transference processing. This can be useful for excluding abnormal or corrupted data points.

***Table B–43 DT_LOC_WK_EXCL_STG***

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| LOC_EXT_KEY | VARCHAR2(80) | External ID for the location. | Y |
| WK_END_DT | DATE | The week end date of a week to be excluded. | Y |
| UPDT_CODE | VARCHAR2(1) | A code to indicate how to update the target table. U=Update/Create, D=Delete existing record. | Y |

## DT_MDL_PROD_EXP_STG

This table is the staging table which provides a list of products that are eligible for processing Model Apply in order to receive product to product demand transferences. The values loaded here should be of the same product hierarchy which DT is associated to work with.

***Table B–44 DT_MDL_PROD_EXP_STG***

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROD_EXT_KEY | VARCHAR2(80) | The external product identifier for the hierarchy to be included in later model apply processing. | Y |

## DT_ASSORT_TRANS_EXP_VW

This view provides an exportable list of assortment multipliers to AIP, so that the impact of assortment changes can influence AIP results.

***Table B–45 DT_ASSORT_TRANS_EXP_VW***

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| LOC_EXT_KEY | VARCHAR2(80) | The external ID for the store location. | N |
| PROD_EXT_KEY | VARCHAR2(80) | The external ID for the product. | N |
| REPL_PROD_EXT_KEY | VARCHAR2(80) | The external ID for the replacement product. | N |
| TRANSFER_PCT | NUMBER(38,20) | The forecasted rate of sale for this product at this location. | Y |

## DT_NEW_ITEM_ROS_EXP_VW

This view provides an exportable view of new items and their forecasted rate of sale to RDF.

*Table B–46 DT_NEW_ITEM_ROS_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROD_EXT_KEY | VARCHAR2(80) | The external ID for the product. | Y |
| LOC_EXT_KEY | VARCHAR2(80) | The external ID for the store location. | Y |
| FCST_ROS | NUMBER(38,20) | The forecasted rate of sale for this product at this location. | Y |

## DT_NEW_ITEMS_EXP_VW

This view provides a list of items and locations for which the item is newly added to the assortment at the location. It is provided in an exportable view for use by RDF.

*Table B–47 DT_NEW_ITEMS_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROD_EXT_KEY | VARCHAR2(80) | The external ID for the product. | Y |
| LOC_EXT_KEY | VARCHAR2(80) | The external ID for the store location. | Y |
| EFF_START_DT | DATE | The date when this product is considered to be added to the assortment. | Y |

## DT_PROD_LOC_EXCL_STG

This is the staging table for products and locations that should be excluded from DT processing, as if they were out of the assortment.

*Table B–48 DT_PROD_LOC_EXCL_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROD_EXT_KEY | VARCHAR2(80) | External product key. | Y |
| LOC_EXT_KEY | VARCHAR2(80) | External id for the location. | Y |
| WK_FROM_DT | DATE | Week date to start the exclusion at. | Y |
| WK_TO_DT | DATE | Week date to end the exclusion at. | Y |

## DT_SIM_EXP_VW

This view provides an export of product similarities calculated within the DT application for export to Category Management.

*Table B–49 DT_SIM_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROD_EXT_KEY1 | VARCHAR2(80) | The external identifier for one half of the product pair. | Y |
| CUSTSEG_EXT_KEY | VARCHAR2(80) | The external identifier for the customer segment this data relates to. | Y |
| LOC_EXT_KEY | VARCHAR2(80) | The external identifier for the store location this data relates to. | Y |

*Table B–49 (Cont.) DT_SIM_EXP_VW*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROD_EXT_KEY2 | VARCHAR2(80) | The external identifier for the other half of the product pair. | Y |
| PROD_SIM | NUMBER(22,7) | The measurement of how similar the two products are to each other. The values range between values of 0 (completely dissimilar) to values of 1 (completely similar). | Y |
| EFFECTIVE_DT_FROM | DATE | The date this similarity value became effective. | Y |
| EFFECTIVE_DT_TO | DATE | The date this similarity value is effective until. | Y |

## PROCESS_STATUS

This table contains processing status columns for use by components that run and manage and report status.

*Table B–50 PROCESS_STATUS*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER | PK for this table. | N |
| SRVC_TYPE | VARCHAR2(30) | An optional FK for the service type, which these statuses belong to. | Y |
| DESCR | VARCHAR2(40) | Descriptive text to describe the status. | Y |
| FAILURE_FLG | VARCHAR2(1) | A flag to indicate whether the status represents success (N/F) or failure (Y/T). | Y |
| EXCLUSIVE_FLG | VARCHAR2(1) | Not used. Intended to identify statuses that indicate that other related steps are not suitable for simultaneous execution. | Y |

**Notes:**

This table contains the status and the description for any batch processes. These are currently used by the service request framework, as well as the job processing framework, and also by any user batch stage processes.

The SRVC_TYPE is used to distinguish between different Service Types in the event that a custom set of statuses are required. This is for future use and is not currently used.

The FAILURE_FLG and EXCLUSIVE_FLG can be NULL or can be a Y/T value to indicate that it is considered a Failed/Exclusive status, or a N/F value to indicate that it is considered a Successful/Non-Exclusive Status. The intent behind the EXCLUSIVE_FLG is to indicate at what point the status should be considered exclusive with another status. For example, two exclusive processes can both be in a Not Running status, or both be in a Complete status, but they cannot both be in a Running status. This feature is not currently implemented.

## RSE_AGGR_SRVC_CONFIG

This table is used to control varying levels of aggregation needs for different database services.

*Table B–51 RSE_AGGR_SRVC_CONFIG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK column. | N |
| SRVC_ID | NUMBER(10) | FK for the DB Service that uses this configuration. | N |
| ENABLED_FLG | VARCHAR2(1) | A flag to indicate whether this aggregation configuration is enabled (Y) for processing or not (N). | Y |

**Notes:**

This table defines various aggregation database services and indicates whether the service should be enabled or not. The SRVC_ID column defines the database service that uses this configuration. This column should contain a compatible key as found in RSE_SRVC_CONFIG.

The ENABLE_FLG allows for temporarily disabling an aggregation process from running (ENABLED_FLG=N).

## RSE_AGGR_SRVC_CONFIG_LEVELS

This table defines the different hierarchy types and levels that need to have aggregate data created as part of this aggregation configuration.

*Table B–52 RSE_AGGR_SRVC_CONFIG_LEVELS*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| AGGR_SRVC_CONFIG_ID | NUMBER(10) | FK for the aggregation service configuration. | N |
| HIER_TYPE_ID | NUMBER(10) | FK for the hierarchy type. | N |
| HIER_LEVEL_ID | NUMBER(4) | FK for the hierarchy level required. | N |

**Notes:**

This table is used to define what aggregation levels are required for a specified aggregation service (AGGR_SRVC_CONFIG_ID). This AGGR_SRVC_CONFIG_ID should be compatible with data found in RSE_AGGR_SRVC_CONFIG.

The HIER_TYPE_ID and HIER_LEVEL_ID should be valid hierarchy types and levels as defined in RSE_HIER_TYPE and RSE_HIER_LEVEL.

This table should only be configured to perform aggregation for the levels that are required in the application in order to reduce the performance implications of calculating these aggregations on a weekly basis.

## RSE_APP_SOURCE

This table defines the list of applications that can be used as a data source for the application components.

*Table B–53 RSE_APP_SOURCE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(255) | A name of this application. This value is not changeable, as it is a reference point in the code. | Y |
| DESCRIPTION | VARCHAR2(255) | Some descriptive text to better describe the application. | Y |

**Notes:**

This table defines applications that can be integrated with. For example, this application has direct integration code with Retail Analytics, and therefore an entry should be present in this table for Retail Analytics. This configuration enables the application to store integration IDs that are specific to Retail Analytics for the various dimension tables. When support for alternative integration IDs is available, then an appropriate entry must be added to this table, and then this ID could be used in the relevant RSE*SRC_XREF tables. The NAME column in this table is a fixed value, as it is possible to use this NAME to lookup the ID within the application code.

## RSE_BUSINESS_OBJECT_ATTR_MD

This table defines the attributes for business objects and also contains relevant details about where to obtain the data for this attribute from external table sources.

*Table B–54 RSE_BUSINESS_OBJECT_ATTR_MD*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this object. | N |
| BUSINESS_OBJECT_MD_ID | NUMBER | FK for the business object that this attribute belongs to. | N |
| BUSINESS_OBJECT_DB_SRC_ID | NUMBER(10) | The FK for the database object source table, which contains the data for this attribute. | N |
| NAME | VARCHAR2(50) | A name for the attribute. | N |
| DESCR | VARCHAR2(255) | A more descriptive name for this attribute. | Y |
| SHORT_DB_NAME | VARCHAR2(30) | An alias for this database column name, which can be used as the column name for this attribute value when the attribute is later used in a cross tab query. | Y |
| SRC_COLUMN_NAME | VARCHAR2(30) | The name of the database column that contains the values for this attribute in the source system. | N |
| PROD_GRP_TYP | VARCHAR2(30) | This defines the type of extended attribute (for example, ITEMDIFF or ITEMUDA) for this attribute. This value should match prod_grp_type in one of the W_RTL_ITEM_GRP#_D tables. | Y |
| ATTR_EXT_CODE | VARCHAR2(255) | This column contains the value of the FLEX_ATTRIB_1_CHAR column for complex attributes. It defines the Attribute code for this attribute. | Y |

*Table B–54 (Cont.) RSE_BUSINESS_OBJECT_ATTR_MD*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| TL_DOMAIN_CODE | VARCHAR2(255) | For attributes that are translated, this domain code should be populated with the corresponding DOMAIN_CODE as used in the TL lookup table (normally ITEM_UDA or DIFF). A null value indicates no translation of the data is required. | Y |
| TL_JOIN_EXPR | VARCHAR2(255) | This column contains an expression that defines how to join to the domain lookup TL table's Domain_member_code to retrieve the proper translated attribute value. | Y |
| EXCLUDE_FLG | VARCHAR2(1) | A Y/N flag to indicate whether the attribute should be used by any modules (N) or whether it should be excluded from use (Y). | Y |

**Notes:**

The purpose of this table is to define the metadata needed for copying attributes from external data sources so that they can be used in this application. These attributes are not automatically updated and do require manual definition, since the scope of attributes and the mapping of the data can vary.

The BUSINESS_OBJECT_MD_ID should be aligned with values from RSE_BUSINESS_OBJECT_MD, and the values in BUSINESS_OBJECT_DB_SRC_ID should be aligned with values from RSE_BUSINESS_OBJECT_DB_SRC.

The NAME and DESCR columns are customizable according to what the attribute value is.

The SHORT_DB_NAME, SRC_COLUMN_NAME, PROD_GRP_TYPE, ATTR_EXT_CODE, TL_DOMAIN_CODE and TL_JOIN_EXPR are all used to define how to retrieve the data from the referenced BUSINESS_OBJECT_DB_SRC table.

For attributes that are contained in a simple column, the SRC_COLUMN_NAME is the sole means for copying the attribute value to the RSE. For attributes that are stored as Item Differentiators in Retail Analytics, the PROD_GRP_TYPE is populated with a value that triggers the attribute code to use special logic to obtain the data from a different column. These types of records require a value in the ATTR_EXT_CODE to enable the process to find the correct user defined attribute record in RA.

The TL_DOMAIN_CODE and TL_JOIN_EXPR columns are present to assist with means of getting the translation values for these user defined attributes from Retail Analytics' translation table.

The EXCLUDE_FLG is used to disable the use of the attribute without requiring it to be deleted from the metadata.

## RSE_BUSINESS_OBJECT_DB_SRC

This table defines the source database objects from which attributes can be found.

*Table B–55 RSE_BUSINESS_OBJECT_DB_SRC*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| DB_OBJECT_NAME | VARCHAR2(30) | The name of the source database table, from which we copy attributes from. | N |
| KEY_COLUMN_NAME | VARCHAR2(255) | The name of the PK column for the source table. | N |
| DB_OBJECT_ALIAS | VARCHAR2(30) | An alias for this table. This alias can be used in the various FILTER_CLAUSE, GROUP_BY_CLAUSE, ORDER_BY_CLAUSE columns of this table. | Y |
| FILTER_CLAUSE | VARCHAR2(2000) | A where clause that can applied to the source table to reduce the rows to be retrieved. The WHERE keyword is not expected in this column. | Y |
| GROUP_BY_CLAUSE | VARCHAR2(255) | A group by expression that can be used when obtaining the attributes from this table. | Y |
| ORDER_BY_CLAUSE | VARCHAR2(255) | An order by clause that can be specified so that data retrieval is ordered. | Y |
| APP_SOURCE_ID | NUMBER(10) | FK to the application source table which provides this attribute data. | N |

**Notes:**

This table's purpose is to define the source tables that attribute values to this application. This metadata is used by the attribute loading process to copy those attribute values from these attribute source tables.

The DB_OBJECT_NAME is a name of a database object that this application has access to and can read from a database synonym. The KEY_COLUMN_NAME defines the column that can be used to uniquely identify the business object that a given attribute value belongs to. This column should exist in the DB_OBJECT_NAME's table. The DB_OBJECT_ALIAS is the alias that can be used in the FILTER_CLAUSE column.

The FILTER_CLAUSE enables the definition of any rules for filtering out records that should not be used. This can be helpful if deleted rows need to be ignored. The GROUP_BY_CLAUSE is used when a source table contains multiple attribute columns to be obtained. In this cause the GROUP_BY_CLAUSE should contain the KEY_COLUMN, plus the alias of ATTR_VALUE,ATTRIBUTE, that the code uses to pivot the multiple columns into multiple rows.

The APP_SOURCE_ID column should be related to a row in the RSE_APP_SOURCE table.

## RSE_BUSINESS_OBJECT_MD

This table identifies the distinct business objects that are handled within the application (such as product, location, customer, and consumer).

*Table B–56 RSE_BUSINESS_OBJECT_MD*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER | PK for this table. | N |
| NAME | VARCHAR2(80) | Internal name of the business object. This value is not changeable. | Y |
| DESCR | VARCHAR2(255) | Descriptive text for the business object. | Y |
| HIER_TABLE_NAME | VARCHAR2(30) | Name of the table that contains the hierarchy for this object. | Y |
| ATTR_TABLE_NAME | VARCHAR2(30) | Name of the table that contains attribute data for this object. | Y |
| XREF_TABLE_NAME | VARCHAR2(30) | Name of the table that contains the integration cross reference ids for external systems. | Y |
| TC_TABLE_NAME | VARCHAR2(30) | Name of the table that contains a transitive closure representation of the hierarchy. | Y |
| LOCAL_ID_NAME | VARCHAR2(30) | The name of the column that contains the internal ID in all tables other than the table specified in HIER_TABLE_NAME (which always uses ID as its internal ID column). | Y |
| EXT_ID_NAME | VARCHAR2(30) | Name of the column that contains the external ID for external systems in the table mentioned in the XREF_TABLE_NAME column. | Y |

**Notes:**

This table defines the types of business objects that the application interacts with. The NAME column is a fixed value that is referenced by application code. This NAME should be unique. The HIER_TABLE_NAME, ATTR_TABLE_NAME, and XREF_TABLE_NAME and TC_TABLE_NAME defines the tables that contain the hierarchy data in this database schema. Not all hierarchies require an Attribute table (ATTR_TABLE_NAME).

The purpose of the columns that specify column names in this table is to assist with code that performs dynamic updating of this data.

## RSE_CONFIG

This table contains configuration names and their values for various settings that can be changed to affect the operation of the application.

*Table B–57 RSE_CONFIG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| APPL_CODE | VARCHAR2(20) | An identifier for the application that this configuration affects. The application code represents a generic configuration that affects all applications. | N |
| PARAM_NAME | VARCHAR2(80) | The name of the configuration. | N |
| PARAM_VALUE | VARCHAR2(255) | The value for this configuration. | Y |

*Table B–57 (Cont.) RSE_CONFIG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| CONFIGURABLE_FLG | VARCHAR2(1) | A flag to indicate whether this configuration is changeable after the application has been installed and initially configured (Y) or whether it is not expected to be changed once the application has been used (N). | N |
| DESCR | VARCHAR2(2000) | A description for the configuration. | Y |
| UPDATEABLE_FLG | VARCHAR2(1) | A flag to indicate whether this configuration value can be updated by the application code (Y) or not (N). | Y |

**Notes:**

This table is used to store configurable keys and their values for use by the application. The APPL_CODE is expected to be one of the application identifiers for the various modules (for example, CDT, CIS, DT, MBA, RSE, and ASO). The application code indicates that the code is applicable available for use by all application modules, while other values should be limited in scope to only that application module.

The PARAM_NAME is the key by which the application looks up a configuration value. This value is a fixed value used throughout the code in order to obtain the value that is contained in the PARAM_VALUE column. This value is adjustable on a per installation basis and should be reviewed and adjusted as required so that it affects the application in the manner that is intended. Some configuration values are not expected to change once the application has completed its configuration stage and has started to run. These types of values are identifiable via the CONFIGURABLE_FLG. Values with a Y can be changed at any time, while values with a N cannot be changed without risk of breaking existing data in the application.

The UPDATEABLE_FLG identifies whether the application has the ability to adjust the PARAM_VALUE as part of the application processes. Some parameters may need to record a record that signifies when something was done, or maybe what the value was processed by a process, while there is no suitable table to hold this type of processing value. So it should be expected that parameters with an UPDATEABLE_FLG=Y can change during the course of the application processes.

## RSE_CONFIG_CODE

This table contains configuration values for configurations that can have different configuration values, depending on another value. For example, if a configuration is needed for a default error tolerance, but department 1 needs 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.

*Table B–58 RSE_CONFIG_CODE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| APPL_CODE | VARCHAR2(20) | An identifier for the application that this configuration affects. The application code represents a generic configuration that affects all applications. | N |
| PARAM_NAME | VARCHAR2(80) | The name of the configuration. | N |
| PARAM_CODE | VARCHAR2(80) | A code that enables specification of a configuration value for a specific data value. | N |

*Table B–58 (Cont.) RSE_CONFIG_CODE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PARAM_VALUE | VARCHAR2(255) | The value for this configuration. | Y |
| CONFIGURABLE_FLG | VARCHAR2(1) | A flag to indicate whether this configuration is changeable after the application has been installed and initially configured (Y) or whether it is not expected to be changed once the application has been used (N). | N |
| UPDATEABLE_FLG | VARCHAR2(1) | A flag to indicate whether this configuration value can be updated by the application code (Y) or not (N). | Y |
| DESCR | VARCHAR2(2000) | A description for the configuration. | Y |

**Notes:**

This table contains configuration values for configurations that can have different configuration values, depending on another value. For example, if a configuration is needed for a default error tolerance, but department 1 needs 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.

The APPL_CODE and PARAM_NAME values in this table are expected to also exist in the RSE_CONFIG table. Normally, there should be a PARAM_VALUE in the RSE_CONFIG table, which is used as the default value for this configuration. Only if there is a PARAM_CODE that matches the specified PARAM_CODE when the configuration is queried, will the PARAM_VALUE be retrieved from this table. Otherwise, the PARAM_VALUE will be retrieved from the RSE_CONFIG table.

Some configuration values are not expected to change once the application has completed its configuration stage and has started to run. These types of values are identifiable via the CONFIGURABLE_FLG. Values with a Y can be changed at any time, while values with a N cannot be changed without risk of breaking existing data in the application.

The UPDATEABLE_FLG identifies whether the application is allowed to adjust the PARAM_VALUE as part of the application processes. Some parameters may need to record a record that signifies when something was done, or maybe what the value was processed by a process, but no suitable table exists to hold this type of processing value. So it should be expected that parameters with an UPDATEABLE_FLG=Y can change during the course of the application processes.

## RSE_CONSEG

This table contains consumer segments and interfaces directly with Retail Analytics.

*Table B–59 RSE_CONSEG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table | N |
| NAME | VARCHAR2(30) | A short name for the consumer segment. | N |
| DESCR | VARCHAR2(255) | A more descriptive name for the consumer segment. | Y |
| CONSEG_EXT_KEY | VARCHAR2(80) | The external key to use when interfacing with external systems. | N |

*Table B–59 (Cont.) RSE_CONSEG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| CONSEG_EXT_CODE | VARCHAR2(80) | The externally visible code that can be shown in any UI components to identify the consumer segment. | Y |
| CONSEG_EXT_ID | NUMBER(20) | The PK for the consumer segment as found in the external system. | N |
| CURRENT_FLG | VARCHAR2(1) | A flag to indicate whether this row is the most recent row (Y) for a given consumer segment, or if it a historical row (N). | Y |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether the row is considered deleted (Y) or not (N). | Y |
| EFF_FROM_DT | DATE | The starting date on which this record is effective on. | Y |
| EFF_TO_DT | DATE | The ending date that this record is effective until. | Y |

## RSE_DISPLAY_FORMAT_STG

This is the staging table used to load display format rules used throughout the UI.

*Table B–60 RSE_DISPLAY_FORMAT_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| APPL_CODE | VARCHAR2(3) | Application code That this display format belongs to. | Y |
| DESCR | VARCHAR2(255) | Description of this display format. | Y |
| EXT_KEY | VARCHAR2(80) | An external identifier to identify this display format entry. This value should be unique within the specified APPL_CODE. | Y |
| CURRENCY_FLG | VARCHAR2(1) | A Y/N flag to indicate whether this display format should include formatting specific to currency (Y) or not (N). | Y |
| PCT_FLG | VARCHAR2(1) | A Y/N flag to indicate whether this display format should include formatting conventions used to display percentages (Y) or not (N). | Y |
| NUM_DEC_DIGITS | NUMBER(10) | The number of digits to display after the decimal point. Only applicable for numeric fields, NULL for all others. | Y |
| ROUNDING_MODE | VARCHAR2(30) | This column defines the rounding approach to use when rounding numeric values. Possible values include: UP, DOWN, CEILING, FLOOR, HALF_UP, HALF_DOWN, HALF_EVEN. | Y |

## RSE_EXCLUSIVE_PROC_TASK_XREF

This table is similar in concept to RSE_EXCLUSIVE_PROC_XREF; however, this table is for tasks instead of for entire processes. The table defines a set of tasks that are exclusive to each other and that should not be running at the same time.

**Notes:**

This table is not currently in use. It is expected to have references to two instances of the RSE_PROC_TASK_TMPL table in order to indicate that a task cannot be run at the same time another task is running. The pair of PROC_TASK_ID1 and PROC_TASK_ID2 columns are expected to be unique.

## RSE_EXCLUSIVE_PROC_XREF

This table is intended to define exclusive processes.

**Notes:**

This table is not currently in use. It is expected to have references to two instances of the RSE_PROC_TMPL table in order to indicate that a process cannot be run at the same time another process is running. The pair of PROC_ID1 and PROC_ID2 columns are expected to be unique.

## RSE_FAKE_CUST_STG

This staging table specifies customers who are considered to be fake customers. A fake customer is a customer who purchases too many transactions to be considered a single customer. Examples are generic store cards.

***Table B–61 RSE_FAKE_CUST_STG***

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| CUSTOMER_NUM | VARCHAR2(80) | The customer ID to be updated. | Y |
| FAKE_CUST_FLG | VARCHAR2(1) | A flag to indicate whether the customer should be identified as fake (Y) or not (N). | Y |

**Notes:**

This interface allows a customer to manually define the fake customers. This interface can be used instead of (or in additional to) the automated routine that is provided to automatically detect fake customers. A fake customer is a customer who purchases an unusual number of transactions and therefore cannot be an actual person shopping individually. Fake customers must be removed from processing so that the unusual buying patterns do not interfere with the analysis of customer purchase decisions.

The CUSTOMER_NUM should be related to the W_PARTY_PER_D.CUSTOMER_NUM column.

This interface can overwrite the FAKE_CUST_FLG value for existing rows, while the automated process cannot. Therefore, if a customer is detected as a fake customer by the automated detection routine, this loader can be used to signify that the customer is not fake, and then on subsequent executions of the automated process, the customer will not be identified as a fake customer.

## RSE_HIER_LEVEL

This table defines the various levels for all the hierarchies.

*Table B–62 RSE_HIER_LEVEL*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(4) | ID for the level. The highest level is 1, and child levels have sequentially increasing values within each Hierarchy Type. | N |
| HIER_TYPE_ID | NUMBER(10) | FK to the Hierarchy Type. | N |
| DESCR | VARCHAR2(80) | Description for this level. For hierarchies that are copied from other systems, this value should match the value in the source system. | Y |
| SRC_HIER_TYPE_ID | NUMBER(10) | For alternate hierarchies, this is a FK for the primary hierarchy that this hierarchy is associated with. | Y |
| SRC_HIER_LEVEL_ID | NUMBER(4) | For alternate hierarchies, this is the level of the other hierarchy that this level is aligned with. This is the point at which the two hierarchies merge. All levels below this hierarchy are common between the two hierarchies. | Y |
| LEAF_NODE_FLG | VARCHAR2(1) | A flag to indicate if this is the lowest level of the hierarchy (Y) or not (N). | N |

**Notes:**

For each hierarchy, it is necessary to have a list of the levels that belong to the hierarchy. The levels are fixed for the hierarchy type (HIER_TYPE_ID) and cannot be of variable depth. The highest ID for a given hierarchy type (HIER_TYPE_ID) should be the leaf level (LEAF_NODE_FLG=Y).

The HIER_TYPE_ID should be defined in the RSE_HIER_TYPE table. Within a given HIER_TYPE_ID, the IDs should range from 1 (for the highest level of the hierarchy), to the lowest level of the hierarchy, which should have the highest ID value. No gaps are allowed in the levels within a hierarchy type.

For non-primary hierarchies, it is possible to define a hierarchy that converges with the primary hierarchy. The point at which this happens should be updated so that the SRC_HIER_LEVEL_ID and SRC_HIER_TYPE_ID contain the appropriate level and hierarchy type where the data aligns with the primary hierarchy.

## RSE_HIER_TYPE

This table hosts the catalog of hierarchies that are available within the application. It identifies the business objects for which the hierarchy applies. It also identifies the primary hierarchy for that object.

*Table B–63 RSE_HIER_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(255) | The name of the hierarchy. | Y |
| DESCR | VARCHAR2(255) | A description for this hierarchy. | Y |

*Table B–63 (Cont.) RSE_HIER_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| LEAF_NODE_LEVEL | VARCHAR2(80) | The level number of the hierarchy that contains the lowest level of data. This should be the maximum ID in the RSE_HIER_LEVEL table for this hierarchy type. | Y |
| PRIMARY_FLG | VARCHAR2(1) | A flag to indicate whether this is the primary hierarchy (Y) for a given business object or not (N). | Y |
| BUSINESS_OBJECT_MD_ID | NUMBER | FK to the RSE_BUSINESS_OBJECT_MD that defines the business object this hierarchy can be used with. | N |

**Notes:**

The purpose of this table is to define all the different hierarchies that are available throughout the system. For a given business object (BUSINESS_OBJECT_MD_ID), only one hierarchy should be considered the primary hierarchy (PRIMARY_FLG=Y). All others should not be primary (PRIMARY_FLG=N). The highest number of the RSE_HIER_LEVEL that relates to this hierarchy type should be defined in the LEAF_NODE_LEVEL column.

## RSE_LIKE_LOC_STG

This is the staging table used to load the like stores for CM Group or Category.

*Table B–64 RSE_LIKE_LOC_CFG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| LOC_EXT_KEY | VARCHAR2(80) | External ID for the store location. | Y |
| LIKE_LOC_EXT_KEY | VARCHAR2(80) | External ID for the like store location. | Y |
| PROD_HIER_TYPE_NAME | VARCHAR2(255) | The name of the product hierarchy type associated with this store. | Y |
| PROD_EXT_KEY | VARCHAR2(80) | The external key to identify the product hierarchy this like store relates to. | Y |
| WEIGHT | NUMBER(9,4) | Weight of the like store associated with the store. | Y |
| EFFECTIVE_START_DT | DATE | The date of the start of the effective period. | Y |
| EFFECTIVE_END_DT | DATE | The date of the end of the effective period. | Y |
| ACTIVE | VARCHAR2(1) | This is a Y/N flag to indicate whether this like store mapping is valid or not. | Y |
| NEW_FLG | VARCHAR2(1) | This is a Y/N flag to indicate whether this store is new or existing store [poor history]. | Y |

## RSE_LOAD_SRVC_CFG

This table defines a data loader that is available for execution through the data loading framework.

*Table B–65 RSE_LOAD_SRVC_CFG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this loader. | N |
| NAME | VARCHAR2(50) | The name of the data loader as referenced by external invocations of the loader. | N |
| DESCR | VARCHAR2(250) | Descriptive text that describes this data loader. | N |
| IMPL_NAME | VARCHAR2(30) | The name of the implementation that performs this data loading. | N |
| CREATED_ON_DT | DATE | The date this data loader was created. | Y |
| CHANGE_ON_DT | DATE | The date this record was last changed. | Y |

**Notes:**

This table contains a list of database loader services, along with the name of the code that provides the implementation for this data loading. If necessary, the IMPL_NAMEs can be extended so that additional data loading rules are implemented. Then the name of the database type in IMPL_NAME can be changed to that name. All of the data types mentioned in IMPL_NAME must be descendants of the RSE_LOAD_SRVC_T database type.

The NAME value is a fixed value that other parts of the application reference these data loaders by and therefore is not changeable. These names should be unique.

## RSE_LOAD_VALDT_RULES_CFG

This table defines the validation rules that a data loader performs, along with some configurable options that affect rows that fail this validation.

*Table B–66 RSE_LOAD_VALDT_RULES_CFG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this validation rule. | N |
| NAME | VARCHAR2(50) | The validation name used in the data loader to reference this validation rule. | N |
| LOAD_SRVC_CFG_ID | NUMBER(10) | FK for the data loader service configuration that uses this validation rule. | N |
| LOAD_VALDT_TYPE | VARCHAR2(1) | FK for the validation type (RSE_LOAD_ VALDT_TYPE). | N |
| DESCR | VARCHAR2(250) | A descriptive message that appears when rows do not pass the validation rule. | N |
| MAX_PCT_ERRORS | NUMBER(3) | An integer representation of the percentage of rows that can fail this validation rule before the data loader aborts processing the remaining the good rows. | Y |
| MAX_NUM_ERRORS | NUMBER(10) | The maximum number of rows that can fail this validation rule before the data loader aborts the loading of the remaining good rows. | Y |
| CREATED_ON_DT | DATE | Date this validation rule was created. | Y |
| CHANGE_ON_DT | DATE | Date this validation rule was last changed. | Y |

**Notes:**

This table defines validation rules that are associated with a data loading database service type. The LOAD_SRVC_CFG_ID should align with a row in the RSE_LOAD_SRVC_CFG table. The NAME column in this table is a fixed name that the application code uses to look up the validation rule, so these names cannot be changed. This NAME column must be unique.

The MAX_PCT_ERRORS and MAX_NUM_ERRORS are used to specify a threshold for ignoring failed validation rules. If the number of records that fail the validation rules are below the threshold, then the bad records are not included in the data loading process. Otherwise, the data loader will abort its data loading, and the bad records can be found in the appropriate *_BAD table.

## RSE_LOAD_VALDT_TYPE

This table defines the different validation types available in a data loader.

*Table B–67 RSE_LOAD_VALDT_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| LOAD_VALDT_TYPE | VARCHAR2(1) | PK for this table (R=Record, C=Complex record, T=Table). | N |
| DESCR | VARCHAR2(50) | A description for this validation type. | N |

**Notes:**

This table defines the list of data loading validation types, available for use by the data loading validations. This table helps provide better reporting of validation errors.

The LOAD_VALDT_TYPE column is referenced via the application code, so the values cannot be changed without impacting the application code.

## RSE_POST_PROC_TMPL

This table is used to define a template for post-processing tasks for a given process task. This table is not currently in use.

**Notes:**

This table is not currently in use, although future releases will use this to define post-processing steps that should be invoked after a process has completed. When the process finishes with a particular status, this table enables different post-processing routines to be executed. As a result, it is possible to invoke a routine to signify to an external entity that a process has completed successfully or that the process failed.

## RSE_PROC_TASK_PREREQ_TMPL

This table defines prerequisites between process tasks. This table is not currently in use.

**Notes:**

This table is not currently in use by the system and is reserved for future use. It can be used to define prerequisite rules for tasks in order to make sure that tasks can be run in a asynchronous manner, with rules to prevent the tasks from running before a prior task has completed.

## RSE_PROC_TASK_TMPL

This table defines templates for processing tasks that are used by the applications.

*Table B–68 RSE_PROC_TASK_TMPL*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| PROC_ID | NUMBER(10) | FK to the RSE_PROC_TMPL table. | N |
| NAME | VARCHAR2(80) | A name for this task. | N |
| DESCR | VARCHAR2(255) | A descriptive name for this processing task. | Y |
| STEP_NUM | NUMBER(4) | A sequential step number for this task, relative to its peer tasks. | Y |
| PARENT_ID | NUMBER(10) | FK to a parent task in this same table. | Y |
| EXEC_PROD_ID | NUMBER(10) | Not currently used. | Y |
| SRVC_TYPE_ID | NUMBER(10) | Not currently used. | Y |
| SRVC_NAME | VARCHAR2(30) | Not currently used. | Y |
| PKG | VARCHAR2(255) | The name of the package that contains the code to be executed. | Y |
| PROC | VARCHAR(255) | The name of the procedure that is executed for this task. | Y |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether this record is to be considered deleted (T) or not (F). | Y |
| ACTIVE_FLG | VARCHAR2(1) | A flag to indicate whether this task should be considered active (T) or not (F). | Y |
| MAX_RUN_TIME_MIN | NUMBER(8) | Not currently used. | Y |
| MAX_FAILURES | NUMBER(8) | Not currently used. | Y |
| RESTARTABLE_FLG | VARCHAR2(1) | Not currently used. | Y |

**Notes:**

This table defines a template of the processing tasks for a batch process. There can be a mixture actual processing tasks, and also of entries whose whole purpose is just to create a hierarchy of tasks. By having a hierarchy of tasks, it allows for concurrent operations to be performed at various points through the work flow, while at the same time, allowing dependencies to be completed before continuing with other tasks.

The ID in this table is a fixed ID and is used to associate with configuration entries in other parts of the application. The STEP_NUM indicates the sequence in which tasks run, within the specified PARENT_ID. The STEP_NUM should be unique within a given PROC_ID, PARENT_ID.

The PKG column contains a value which associates with other application code to signify what code or type of code is to be run for this step.

## RSE_PROC_TMPL

This table defines processing templates for asynchronous or synchronous invocable from Java applications.

*Table B–69 RSE_PROC_TMPL*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROC_ID | NUMBER(10) | PK for this table | N |
| APPL_ID | VARCHAR2(8) | Application identifier for this process. | Y |
| NAME | VARCHAR2(80) | Name of the process, as referenced by application code. | Y |
| DESCR | VARCHAR2(255) | Descriptive text for this process. | Y |
| ASYNC_FLG | VARCHAR2(1) | A flag to indicate whether this process is asynchronous (T) or synchronous (F). | Y |
| THREAD_LIMIT | NUMBER(4) | Not currently used. | Y |
| THREAD_GRP | VARCHAR2(8) | Not currently used. | Y |
| EXCLUSIVE_GRP | VARCHAR2(8) | Not currently used. | Y |
| ACTIVE_FLG | VARCHAR2(1) | A flag to indicate whether this process is active for use (T) or not (F). | Y |
| DELETE_FLG | VARCHAR2(1) | A flag to indicate whether this process is considered deleted (T) or not (F). | Y |
| MAX_RUN_TIME_MIN | NUMBER(8) | Not currently used. | Y |
| MAX_FAILURES | NUMBER(8) | Not currently used. | Y |
| RESTARTABLE_FLG | VARCHAR2(1) | Not currently used. | Y |

**Notes:**

This table defines the templates of processes that can be invoked from the applications.

The application ID is a short identifier to indicate which application owns the task. Currently available values are CIS, CDT, DT, MBA, and ASO.

The NAME column is the name that the application uses for lookup in order to find a run-able process thread. This column must be unique within each application.

The IDs in this table are also predefined and cannot be changed. The application may do lookups for tasks either by PROC_ID or by NAME.

## RSE_PROD_ATTR_GRP_VALUE_STG

This is the staging table used to load the associations of CM Groups to product attributes.

*Table B–70 RSE_PROD_ATTR_GRP_VALUE_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROD_HIER_TYPE_NAME | VARCHAR2(255) | The name of the product hierarchy type associated with this Attribute Group value. | Y |
| PROD_EXT_KEY | VARCHAR2(80) | The external key to identify the product hierarchy this product attribute group value relates to. | Y |
| ATTR_SHORT_DB_NAME | VARCHAR2(30) | The short name for the attribute which this product attribute group is related to. | Y |
| PROD_ATTR_GRP_EXT_KEY | VARCHAR2(80) | The external key to uniquely identify the product attribute group. | Y |
| PROD_ATTR_GRP_NAME | VARCHAR2(255) | The name for the product attribute group. | Y |

*Table B–70 (Cont.) RSE_PROD_ATTR_GRP_VALUE_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROD_ATTR_GRP_DESCR | VARCHAR2(255) | The description for the product attribute group. | Y |
| PROD_ATTR_VALUE_KEY | VARCHAR2(255) | The external key to uniquely identify the product attribute group value. | Y |
| PROD_ATTR_VALUE_NAME | VARCHAR2(255) | The name for the product attribute group value. | Y |
| PROD_ATTR_VALUE_DESCR | VARCHAR2(255) | The description for the product attribute group value | Y |
| FUNC_ATTR_FLG | VARCHAR2(1) | 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). | Y |

**Notes:**

This table defines the set of attributes and attribute values for those attributes.

The only NULLABLE columns for this interface are the PROD_ATTR_GRP_DESCR and PROD_ATTR_VALUE_DESCR columns.

The data should be joinable to RSE_PROD_SRC_XREF via the PROD_EXT_KEY, where the LEAF_FLG = Y. The PROD_HIER_TYPE_NAME should be joinable to the RSE_HIER_TYPE table on the Name column.

The values in the PROD_ATTR_GRP_EXT_KEY must be uniquely assigned to a PROD_EXT_KEY.

## RSE_PROD_ATTR_VALUE_XREF_STG

This table contains the cross reference of product attribute values to the CM Group Attribute Value Groups.

*Table B–71 RSE_PROD_ATTR_VALUE_XREF_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PROD_ATTR_VALUE_KEY | VARCHAR2(255) | External key to identify the product attribute group value this xref is for. | Y |
| MIN_ATTR_NUM_VALUE | NUMBER(22,5) | Minimum number value for this xref. Inclusive of this value. | Y |
| MAX_ATTR_NUM_VALUE | NUMBER(22,5) | Maximum number value for this xref. This value is not inclusive in this xref. | Y |
| ATTR_STRING_VALUE | VARCHAR2(255) | An attribute string value to associate with this attribute group value. | Y |
| MIN_ATTR_DATE_VALUE | DATE | Minimum date value to associate with this attribute group value. This value is inclusive. | Y |
| MAX_ATTR_DATE_VALUE | DATE | Maximum attribute value to include for this attribute group value. This value is not inclusive in this range. | Y |
| ATTR_VALUE_EXT_CODE | VARCHAR2(255) | An external attribute code to associate with this attribute group value. | Y |

**Notes:**

This table must be joinable to the RSE_PROD_ATTR_GRP_VALUE_STG table.

This interface must be unique across all columns of this table.

The PROD_ATTR_VALUE_KEY must be joinable to data that is provided by the related RSE_PROD_ATTR_GRP_VALUE_STG table.

The MIN_ATTR_NUM_VALUE/MAX_ATTR_NUM_VALUE columns must be provided as a set, when one is provided.

The MIN_ATTR_DATE_VALUE/MAX_ATTR_DATE_VALUE columns must be provided as a set, when one is provided.

Every PROD_ATTR_VALUE_KEY that is provided by the RSE_PROD_ATTR_GRP_VALUE_STG should have some rows provided in this interface so that attribute values can be found and associated with this attribute value.

The ATTE_VALUE_EXT_CODE is expected to be joined with RA's W_RTL_ITEM_GRP1_D.

One set of value columns should be provided per row (that is, MIN_ATTR_NUM_VALUE and MAX_ATTR_NUM_VALUE but not at the same time as specifying a value for ATTR_STRING_VALUE).

## RSE_SLS_PR_LC_CS_WK_STG

This is the staging table to load aggregate sales data for a product, location, customer segment, and week.

*Table B–72 RSE_SLS_PR_LC_CS_WK_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| WK_END_DT | DATE | The date of the end of the fiscal week. | Y |
| PROD_EXT_KEY | VARCHAR2(80) | External ID for the Product. | Y |
| LOC_EXT_KEY | VARCHAR2(80) | External ID for the store location. | Y |
| CUSTSEG_EXT_KEY | VARCHAR2(80) | External ID for the customer segment. | Y |
| SLS_QTY | NUMBER(38,20) | Quantity of units sold for this entity while not on promotion. | Y |
| SLS_AMT | NUMBER(38,20) | Global sales amount sold for this entity while not on promotion. | Y |
| PROFIT_AMT | NUMBER(38,20) | Amount of profit for this entity while not on promotion. | Y |
| SLS_PR_QTY | NUMBER(38,20) | The number of units sold that was associated with a promotion. | Y |
| SLS_PR_AMT | NUMBER(38,20) | The global sales currency amount that was associated with a promotion. | Y |
| SLS_PR_PROFIT_AMT | NUMBER(38,20) | Amount of global current profit amount for this entity that was associated with a promotion. | Y |

**Notes:**

The data should be unique for the WK_END_DT, PROD_EXT_KEY, LOC_EXT_KEY, and CUSTSEG_EXT_KEY columns.

The WK_END_DT should be related to historical date that is within the fiscal calendar hierarchy. This interface expects the data to be provided in a weekly aggregate form, according to the definition of the week in the fiscal calendar hierarchy.

The PROD_EXT_KEY should be related to a LEAF_NODE_FLG=Y row in the RSE_PROD_SRC_XREF for the primary product hierarchy.

The LOC_EXT_KEY should be related to a LEAF_NODE_FLG=Y row in the RSE_LOC_SRC_XREF table for the primary location hierarchy.

The CUSTSEG_EXT_KEY should be related to a LEAF_NODE_FLG=Y row in the RSE_CUSTSEG_SRC_XREF table.

The SLS_QTY, SLS_AMT, and PROFIT_AMT columns should represent the sales of the product during the week period that are not attributed to a promotion.

The SLS_PR_QTY, SLS_PR_AMT, and SLS_PR_PROFIT_AMT columns should represent the sales of the product during the week period that are attributed to a promotion.

## RSE_SLS_PR_LC_WK_STG

This is the staging table to load aggregate sales data for a product, location, and week.

*Table B–73 RSE_SLS_PR_LC_WK_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| WK_END_DT | DATE | The date of the end of the fiscal week | Y |
| PROD_EXT_KEY | VARCHAR2(80) | External ID for the Product. | Y |
| LOC_EXT_KEY | VARCHAR2(80) | External ID for the store location. | Y |
| SLS_QTY | NUMBER(38,20) | Quantity of units sold for this entity while not on promotion. | Y |
| SLS_AMT | NUMBER(38,20) | Global sales amount sold for this entity while not on promotion. | Y |
| PROFIT_AMT | NUMBER(38,20) | Amount of profit for this entity while not on promotion. | Y |
| SLS_PR_QTY | NUMBER(38,20) | The number of units sold that was associated with a promotion. | Y |
| SLS_PR_AMT | NUMBER(38,20) | The global sales currency amount that was associated with a promotion. | Y |
| SLS_PR_PROFIT_AMT | NUMBER(38,20) | Amount of global current profit amount for this entity that was associated with a promotion. | Y |

**Notes:**

The data should be unique for the WK_END_DT, PROD_EXT_KEY, and LOC_EXT_KEY columns.

The WK_END_DT should be related to historical date that is within the fiscal calendar hierarchy. This interface expects the data to be provided in a weekly aggregate form, according to the definition of the week in the fiscal calendar hierarchy.

The PROD_EXT_KEY should be related to a LEAF_NODE_FLG=Y row in the RSE_PROD_SRC_XREF for the primary product hierarchy.

The LOC_EXT_KEY should be related to a LEAF_NODE_FLG=Y row in the RSE_LOC_SRC_XREF table for the primary location hierarchy.

The SLS_QTY, SLS_AMT, and PROFIT_AMT columns should represent the sales of the product during the week period that are not attributed to a promotion.

The SLS_PR_QTY, SLS_PR_AMT, and SLS_PR_PROFIT_AMT columns should represent the sales of the product during the week period that are attributed to a promotion.

## RSE_SRVC_CONFIG

This table defines all the database service routines available to be invoked through the database service framework in the application.

*Table B–74 RSE_SRVC_CONFIG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| SRVC_ID | NUMBER(10) | PK for the database service. | N |
| SRVC_TYPE_ID | NUMBER(10) | FK to the service type for this service. | N |
| SRVC_NAME | VARCHAR2(30) | The name used to lookup this database service. | N |
| DESCR | VARCHAR2(255) | Description of what this database service does. | Y |
| SRVC_IMPL_NAME | VARCHAR2(30) | The name of the database object that implements this database service. This must be a descendant of RSE_SRVC_T. | Y |
| SRVC_OWNER | VARCHAR2(30) | The database schema that owns the implementation of this service. | Y |
| CREATED_ON_DT | DATE | Date this record was created. | Y |
| CHANGE_ON_DT | DATE | Date this record was last changed. | Y |

**Notes:**

This table enables the overriding of implementations for any of the database services used by the application. The application code looks for services by the Service Type (RSE_SRVC_TYPE.SRVC_TYPE) and the Service Name (SRVC_NAME). If it is necessary to customize an implementation of any service in this table, it is possible to do so by creating new database types that extend the provided database types, and then it is possible to extend or override the existing implementation. Once a new database type is required, the SRVC_IMPL_NAME can be adjusted to refer to the new database type that performs the same functional results.

All database types defined in this table must extend the RSE_SRVC_T database type and should implement and extend this type as needed.

This table has a FK relationship with the RSE_SRVC_TYPE table (SRVC_TYPE_ID). This table should be unique across the SRVC_TYPE_ID and the SRVC_NAME.

It is also expected that the SRVC_OWNER be a single schema. Future releases will explore the ability of offering accessing code from other schemas.

## RSE_SRVC_TYPE

This table defines the types of database services that a database service can belong to. In order for multiple database services to share the same service type, they should be interchangeable at run time.

*Table B–75 RSE_SRVC_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| SRVC_TYPE_ID | NUMBER(10) | PK for the service type. | N |
| SRVC_TYPE | VARCHAR2(30) | The name that is used to look up the database service type and its related services. | N |
| CREATED_ON_DT | DATE | Date this record was created. | Y |
| CHANGE_ON_DT | DATE | Date this record was change. | Y |

**Notes:**

This table defines the types of database services that a database service can belong to. In order for multiple database services to share the same service type, they should be interchangeable at run time.

This table groups related database services together so that they can be batched together and processed in parallel with each other. For example, this might be helpful if there are four different aggregation routines that all process data in a similar manner and all have the same basic setup of parameters. By creating them all under the same service type, it is possible to queue the execution of all four routines, and the process can then process them all together by the same service request group. In a situation like this, the service request group would likely be the week to be processed, and by running them all in one pass, it enables them to all run the same week at the same time. This can be helpful on database resources, as the database can cache a single week of data and then process all steps using that week at the same time. The alternative approach is be to process each aggregation routine individually, one week at a time. But if this is done, then by the time the next aggregation routine is executed, the weeks data is probably no longer cached and must be read again.

The one limitation that exists with grouping services under the same service type is that the pre-processing and post-processing service request group routines must all be the same between all service implementations. This is normally only the case when the service routines all inherit from a common base service or if the service implementations do not actually have pre-service and post-service request group tasks.

## SO_ALERT

This table hosts the list of possible alerts that can be raised by the application.

*Table B–76 SO_ALERT*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the alert. This value is static. The list of recognized values is provided by the application. | N |
| NAME | VARCHAR2(80) | The alert name. This value should not be used for display within the UI. It has a back end use. | Y |
| DESCR | VARCHAR2(200) | A short alert description. | Y |

*Table B–76 (Cont.) SO_ALERT*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| SEVERITY_ID | NUMBER(10) | Foreign key to so_alert_severity table. This field identifies the severity of the event (such as blocking or informative). | N |
| TYPE_ID | NUMBER(10) | Foreign key to so_alert_type table. This value identifies the type of event/exception that raised the alert (such as constraint or space). | N |
| ALERT_MSG | VARCHAR2(200) | Message that provides more information to the user about the problem. Sanity checker uses this value to provide the user with an action to be taken to fix an identified problem. | Y |

**Notes:**

This is a lookup table that provides alerts supported by the application. It is referenced by other tables.

The application can raise these alerts to flag the occurrence of specific events. These alerts are used to notify the user about specific situations, such as space issues and constraint problems.

## SO_ALERT_SEVERITY

This table contains the list of alert severities supported by the application.

*Table B–77 SO_ALERT_SEVERITY*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the alert severity. This value is static. The list of recognized values is provided by the application. | N |
| NAME | VARCHAR2(80) | Alert severity name. This value should not be used for display within the UI. It has a back end use. | Y |
| DESCR | VARCHAR2(200) | Alert severity description. | Y |

**Notes:**

This is a lookup table that provides alert severities supported by the application. It is referenced by other tables. Values include informative, warning, and blocking.

## SO_ALERT_STATUS

This table contains the list of alert status that are supported by the application.

*Table B–78 SO_ALERT_STATUS*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the alert status. This value is static. The list of recognized values is provided by the application. | N |
| Name | VARCHAR2(80) | Alert status name. This value should not be used for display within the UI. It has a back end use. | Y |
| DESCR | VARCHAR2(200) | Alert status description. | Y |

**Notes:**

This is a lookup table that provides alert status supported by the application.It is referenced by other tables.Values include new, ignored, displayed, and accepted.

## SO_ALERT_TYPE

This table contains the list of alert types supported by the application.

*Table B–79 SO_ALERT_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the alert type. This value is static. The list of recognized values is provided by the application. | N |
| NAME | VARCHAR2(80) | Alert type name. This value should not be used for display within the UI. It has a back end use. | Y |
| DESCR | VARCHAR2(200) | Alert type description. | Y |

**Notes:**

This is a lookup table that provides the list of alert types supported by the application. It is referenced by other tables.Values include constraints, blocking, optimization, mapping, assortment, loading, and export.

## SO_ASSORT_CLUSTER_MEMBER_STG

This staging table is used for stores assigned to a specific cluster for the given assortment.

*Table B–80 SO_ASSORT_CLUSTER_MEMBER_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ASSORTMENT_ID | VARCHAR2(80) | ID that identifies the assortment. It must match an assortment key within the assortment file. | Y |
| LOCATION_KEY | VARCHAR2(80) | This is the external store ID, known and shared across applications. | Y |

*Table B–80 (Cont.) SO_ASSORT_CLUSTER_MEMBER_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| CLUSTER_KEY | VARCHAR2(80) | Internal Cat-Man cluster key. This key must match one of the cluster keys provided within the assortment cluster file. | Y |
| START_DT | DATE | Start date range to be used for the store to retrieve forecast data. The format must be YYYY-MM-DD. | Y |
| END_DT | DATE | End date range to be used for the store to retrieve forecast data. The format must be YYYY-MM-DD. | Y |

**Notes:**

This data is mandatory.

These are the stores delivered within the assortment interface, grouped within clusters. ASO expects to always receive stores grouped within clusters. If stores must be sent individually, a cluster must be created for than single store. Product list is linked directly to a store whenever the assortment type = 2 (Store). The start and end date are only included in this table for assortments delivered at the store level.

## SO_ASSORT_CLUSTER_STG

This is the staging table for assortment placeholder products included within the assortment.

*Table B–81 SO_ASSORT_CLUSTER_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ASSORTMENT_ID | VARCHAR2(80) | ID that identifies the assortment. It must match an assortment key within the assortment file. | Y |
| CLUSTER_KEY | VARCHAR2(80) | Cat-Man cluster key. A value that can be returned from SO to Cat-Man to uniquely identify the cluster_id. | Y |
| CLUSTER_NAME | VARCHAR2(80) | Name associated to the cluster. The end user should recognize this name as the cluster name seen or entered within Cat-Man | Y |
| START_DT | DATE | Start date range to be used for the stores within the cluster to retrieve forecast data. The format must be YYYY-MM-DD | Y |
| END_DT | DATE | End date range to be used for the stores within the cluster to retrieve forecast data. The format must be YYYY-MM-DD | Y |

**Notes:**

This data is mandatory.

These clusters are delivered within the assortment interface. ASO expects to always receive stores grouped within clusters. If the stores need to be sent individually, a cluster must be created for that single store. Product list is linked directly to a cluster whenever the assortment type = 1 (Cluster). Start and end dates are only included for assortments delivered at the cluster level.

## SO_ASSORT_PHPROD_ATTR_STG

This is the staging table for attribute data for assortment placeholder products.

*Table B–82 SO_ASSORT_PHPROD_ATTR_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ASSORTMENT_ID | VARCHAR2(80) | ID that identified the assortment. It must match an assortment key within the assortment file. | Y |
| PLACEHOLDER_PRODUCT_KEY | VARCHAR2(80) | Cat-Man product key for placeholder product specific to the assortment. It must be different from other formalized products. | Y |
| ATTR_NAME | VARCHAR2(50) | Name of the product attribute that is being passed. It must match a known product attribute. | Y |
| ATTR_VALUE | VARCHAR2(50) | Specific value that should be used for the placeholder product/attribute combination. | Y |

**Notes:**

This data is optional.

This table includes only placeholder product attributes. Attribute names must match existing attributes already available within ASO and shared with the other products.

## SO_ASSORT_PHPROD_FINALIZED_STG

This is the staging table that receives finalized assortment placeholder products included within the assortment. This data is used to transform the placeholder name and ID.

*Table B–83 SO_ASSORT_PHPROD_FINALIZED_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ASSORTMENT_SET_ID | VARCHAR2(80) | Unique assortment set ID. This value is used to group together multiple assortments (user requests). | N |
| PLACEHOLDER_PRODUCT_KEY | VARCHAR2(80) | Category Management product key for placeholder product specific to the assortment. Must be different from other formalized products. | N |
| FINALIZED_PRODUCT_NAME | VARCHAR2(80) | Tag that describes the finalized placeholder item. | Y |
| FINALIZED_PRODUCT_KEY | VARCHAR2(80) | This field must match a Product Key Definition in RSE Core. This is the external ID that is known and shared across applications. The like product key must be one of the known products also included within the assortment. | Y |

## SO_ASSORT_PHPROD_LIKE_PROD_STG

If a placeholder product is included in the assortment, it must appear in this feed.

The following is a list of placeholder products included in the assortment. Each placeholder item must be paired with an existing product. ASO uses the next pieces of data from the existing product (like item).

- MSM-type data such as size or other product merchandising information.
- Merchandise hierarchy information (where this product sits in the hierarchy)
- Product attributes (such as what is used for DT calls)
- ASO-only data (sku/store replenishment parameters used in ASO and other SO inputs)

Product Key for placeholder items must always be different than the one for any known product.

*Table B–84 SO_ASSORT_PHPROD_LIKE_PROD_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ASSORTMENT_ID | VARCHAR2(80) | ID that identified the assortment. It must match an assortment key within the assortment file. | Y |
| PLACEHOLDER_PRODUCT_KEY | VARCHAR2(80) | Category Management product key for placeholder product specific to the assortment. It must be different than other formalized products. | Y |
| PLACEHOLDER_PRODUCT_NAME | VARCHAR2(80) | Tag that describes the placeholder item. It is used by the UI to look at product level data. | Y |
| LIKE_PRODUCT_KEY | VARCHAR2(80) | This field must match a Product Key Definition in RSE Core. This is the external ID that is known and shared across applications. The like product key must be one of the known products also included within the assortment. | Y |

**Notes:**

This data is optional if placeholder products are included.

## SO_ASSORT_PRODUCT_STRCLTR_STG

This feed also includes placeholder products.

*Table B–85 SO_ASSORT_PRODUCT_STRCLTR_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ASSORTMENT_ID | VARCHAR2(80) | ID that identifies the assortment. It must match an assortment key within the assortment file. | Y |
| CLUSTER_STORE_KEY | VARCHAR2(80) | This field can be either an assortment cluster key or a location key. The actual value is determined by the assortment type (If Assortment_type = 1 (ClusterAssortment key) and If Assortment_type = 2 (Store Assortment key)). The external store ID must be the one known and shared across applications | Y |

*Table B–85 (Cont.) SO_ASSORT_PRODUCT_STRCLTR_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PRODUCT_KEY | VARCHAR2(80) | This is the external product ID that is known and shared across applications. For placeholder products, this field contains the Category Management placeholder product key that must be different than any known product. | Y |
| IPI_VALUE | NUMBER(18,4) | This value can be NULL if not available. | Y |
| PRIORITY | NUMBER(2) | This field can take four different values, 1 = mandatory, 2 = core, 3 = optional, and -1=dropped. Records with -1 may be filtered out. | Y |

**Notes:**

This data is mandatory.

Placeholder products must also be included within this table. An assortment can be delivered either at the store or cluster level, but not both at the same time. The product key for placeholder products must always be different than the one used for any known product.

## SO_ASSORT_PROLOC_FCST_STG

This feed also includes placeholder products.

*Table B–86 SO_ASSORT_PROLOC_FCST_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ASSORTMENT_ID | VARCHAR2(80) | ID that identifies the assortment. It must match an assortment key within the assortment file. | Y |
| PRODUCT_KEY | VARCHAR2(80) | This is the external ID that is known and shared across applications. For placeholder products this field contains a Category Management placeholder product key. | Y |
| LOCATION_KEY | VARCHAR2(80) | This is the external store ID, known and shared across applications. | Y |
| WEEKLY_PERIOD | DATE | Week start date for which the forecast is provided. | Y |
| DEMAND | NUMBER(18,4) | Forecast demand for the week. | Y |
| ERROR_TERM | NUMBER(18,4) | Not currently in use. Default to NULL. | Y |

**Notes:**

This data is mandatory.

This table receives forecast data for all the products within the assortment, including placeholder products. The forecast must cover the range of dates specified for the cluster or stores.

## SO_ASSORT_PROLOC_PRICECOST_STG

This feed also includes placeholder products.

*Table B–87 SO_ASSORT_PROLOC_PRICECOST_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ASSORTMENT_ID | VARCHAR2(80) | ID that identifies the assortment. It must match an assortment key within the assortment file. | Y |
| PRODUCT_KEY | VARCHAR2(80) | This is the external ID that is known and shared across applications. For placeholder products this field contains a Category Management placeholder product key. | Y |
| LOCATION_KEY | VARCHAR2(80) | This is the external store ID, known and shared across applications. | Y |
| PRICE | NUMBER(18,4) | Product price, single currency unit determined at the configuration level. No multi-currency is allowed. | Y |
| COST | NUMBER(18,4) | Product cost, single currency unit determined at the configuration level. No multi-currency is allowed | Y |

**Notes:**

This data is mandatory.

The data on this table must be delivered at the product/location level. This table must include the corresponding price and cost for placeholder products (if any is included within the assortment).

## SO_ASSORTMENT_FINALIZED_STG

This table contains the details about the assortment location types supported by the application.

*Table B–88 SO_ASSORTMENT_FINALIZED_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ASSORTMENT_SET_ID | VARCHAR2(80) | Unique assortment set ID. This value is used to group together multiple assortments (user requests) | N |
| PRODUCT_CATEGORY_KEY | VARCHAR2(80) | This value has to match a node in merchandise hierarchy. This is the external ID that is known and shared across applications. | Y |
| ASSORT_LABEL | VARCHAR2(80) | This is a user-entered field with a default value. This value is presented within the UI as the familiar label/name recognizable by the user. It can be NULL. | Y |
| TRADE_AREA_LABEL | VARCHAR2(80) | Category Management trading area used to extract the assortment. | Y |

*Table B–88 (Cont.) SO_ASSORTMENT_FINALIZED_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ASSORT_ROLE | VARCHAR2(50) | This is a Category Management data element that should be passed to ASO. ASO shows this value within BI modules. This is relevant for users when they pick the optimization objective function. | Y |
| ASSORT_TACTIC | VARCHAR2(100) | This is a Category Management data element that should be passed to ASO. ASO shows this value within BI modules. This is relevant for user when they pick the optimization objective function. | Y |
| ASSORT_GOAL | VARCHAR2(50) | This is a Cat-Man data element that should be passed to SO. SO shows this value within BI modules. This is relevant for the users when they pick the optimization objective function. | Y |

## SO_ASSORTMENT_LOC_TYPE

This table contains the details about the assortment location types supported by the application.

*Table B–89 SO_ASSORTMENT_LOC_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the assortment location type. This value is static. The list of recognized assortment location types is provided by the application. 1-Cluster Assortment and 2-Store assortment. | N |
| DESCR | VARCHAR2(80) | Assortment location type description. | Y |

**Notes:**

The application supports the delivery of assortment in two different ways, the cluster level or the store level.

## SO_ASSORTMENT_REQ_TYPE

This table contains the list of possible assortment request types that can be supported by the application.

*Table B–90 SO_ASSORTMENT_REQ_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the assortment request type. This value is static. The list of recognized assortment request types is provided by the application. | N |
| DESCR | VARCHAR2(80) | Assortment request type description. | Y |

**Notes:**

This is a lookup table that provides assortment request types that can be supported by the application. It is referenced by other tables. Values include Optimization Request, Finalized Assortment Reporting, and Promotional Assortment.

## SO_ASSORTMENT_STATUS

This table contains the list of possible status for the assortments (for example, Received, Mapped, Optimized, Processed, Accepted, Discarded, POG mapping needs review, Ready for SO, and User Re-mapping).

*Table B–91 SO_ASSORTMENT_STATUS*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ASSORTMENT_STATUS | VARCHAR2(10) | Unique identifier for the assortment status. This value is static. The list of recognized status is provided by the application. | N |
| DESCR | VARCHAR2(80) | Assortment status description. | Y |
| UI_DESCR | VARCHAR2(80) | Assortment status description to be used within the UI. | Y |

**Notes:**

This is a lookup table that provides assortment status. It is referenced by other tables.

## SO_ASSORTMENT_STG

This interface table contains assortment header and general information.

*Table B–92 SO_ASSORTMENT_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | VARCHAR2(80) | System-generated Primary Key that is the internal assortment identifier. | Y |
| PRODUCT_CATEGORY_KEY | VARCHAR2(80) | This value must match a node in merchandise hierarchy. This is the external ID that is known and shared across applications. | Y |
| ASSORT_LABEL | VARCHAR2(80) | This is a user-entered field with a default value. This value is presented within the UI as the familiar label/name recognizable by the user. It can be NULL. | Y |
| TRADE_AREA_LABEL | VARCHAR2(80) | Category Management trading area used to extract the assortment. | Y |
| REQUEST_TYPE | NUMBER(2) | This field can accept two values: 1=Optimization Request and 2=Finalized Assortment Reporting. | Y |
| ASSORT_LOC_TYPE | NUMBER(2) | This field can accept two values: 1= Cluster Assortment and 2=Store Assortment. This field indicates the level at which the assortment is delivered. | Y |
| ASSORT_ROLE | VARCHAR2(50) | This is a Category Management data element that should be passed to ASO. ASO shows this value within BI modules. This is relevant for users when they pick the optimization objective function. | Y |

*Table B–92 (Cont.) SO_ASSORTMENT_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ASSORT_TACTIC | VARCHAR2(100) | This is a Category Management data element that should be passed to ASO. ASO shows this value within BI modules. This is relevant for users when they pick the optimization objective function. | Y |
| ASSORT_GOAL | VARCHAR2(50) | This is a Category Management data element that should be passed to ASO. ASO shows this value within BI modules. This is relevant for users when they pick the optimization objective function. | Y |
| ASSORTMENT_SET_ID | VARCHAR2(80) | Unique assortment set ID. This value is used to group together multiple assortments (user requests). | Y |

**Notes:**

This data is mandatory.

This staging table is used to receive assortment data from external sources. Each assortment provided must have a unique assortment_id.

## SO_BAY_FIXTURE_SHELF_STG

This table describes the shelf layout in a Fixture. It is used for Shelf Fixture only.

*Table B–93 SO_BAY_FIXTURE_SHELF_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BAY_KEY | VARCHAR2(80) | Bay external ID. Bay is a direct dependent of the planogram. | Y |
| FIXTURE_KEY | VARCHAR2(80) | Fixture external ID. Fixture is a direct dependent of the Bay. | Y |
| SHELF_KEY | VARCHAR2(80) | Shelf external ID. Shelf is a direct dependent of shelf fixture. | Y |
| POS_X | NUMBER(18,4) | Position of the shelf on the X axis. Origin point: bottom, left, back (within the fixture). | Y |
| POS_Y | NUMBER(18,4) | Position of the shelf on the Y axis. Origin point: bottom, left, back (within the fixture). | Y |
| POS_Z | NUMBER(18,4) | Position of the shelf on the Z axis. Origin point: bottom, left, back (within the fixture). | Y |

**Notes:**

This table has the information about the specific shelves that are included within a given shelf fixture. This table is only populated for planograms that include shelf fixtures; data is not available within this table for pegboard or freezer chest fixtures.

## SO_BAY_FIXTURE_STG

This table receives the fixture layout within a Bay. The fixture can be Shelf, Pegboard, or Freezer.

*Table B–94 SO_BAY_FIXTURE_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BAY_KEY | VARCHAR2(80) | Bay external ID. Bay is a direct dependent of the planogram. | Y |
| FIXTURE_KEY | VARCHAR2(80) | Fixture external ID. Fixture is a direct dependent of the bay. | Y |
| POS_X | NUMBER(18,4) | Position of the fixture on the X axis. Origin point: bottom, left, back (within the bay). | Y |
| POS_Y | NUMBER(18,4) | Position of the fixture on the Y axis. Origin point: bottom, left, back (within the bay). | Y |
| POS_Z | NUMBER(18,4) | Position of the fixture on the Z axis. Origin point: bottom, left, back (within the bay). | Y |

**Notes:**

This table contains the information about the different fixtures that are assigned to every planogram bay. There can be multiple fixtures within a bay; each fixture within a bay keeps the same layout from left to right.

## SO_BI_CHART_TYPE

This table contains the list of possible BI chart types that are supported by the application.

*Table B–95 SO_BI_CHART_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| CHART_TYPE | VARCHAR2(15) | Unique identifier for the BI chart type. This value is static. The list of recognized values is provided by the application. | N |
| DESCR | VARCHAR2(30) | BI chart type description. | N |

**Notes:**

This is a lookup table that provides BI chart types that can be supported by the application. It is referenced by other tables. Values include histogram, chart, and pie.

## SO_BI_ELEMENT

This table contains configuration metadata for business intelligence elements.

*Table B–96 SO_BI_ELEMENT*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BI_ELEMENT | VARCHAR2(30) | Unique identifier for the BI element. This value is static. The list of recognized values is provided by the application. This ID is directly referenced by the UI. | N |
| DESCR | VARCHAR2(50) | BI element description. This value is frequently used by the UI as a label. | N |
| UI_MODULE | VARCHAR2(30) | Foreign key from so_ui_module. This value identifies the BI module that makes use of the element. | N |

*Table B–96 (Cont.) SO_BI_ELEMENT*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| DEFAULT_FLG | VARCHAR2(1) | Y/N Indicator that is used to identify the element within a section that should be used by the UI as a default value. | N |
| BI_SECTION | VARCHAR2(15) | Foreign key from so_bi_section. This value identified the BI section that uses this element. | N |
| DISPLAY_SEQ | NUMBER(4) | There could be multiple BI elements within the same BI section; the values might be used within UI list of values. This field indicates the sequence in which the value should be presented to the user. | N |

**Notes:**

This is a lookup and metadata table that tracks the different business intelligence elements that require some metadata from the database. It is referenced by other tables and use by the UI to build BI sections.

## SO_BI_ELEMENT_CHART

This table that contains the metadata to configure BI element charts.

*Table B–97 SO_BI_ELEMENT_CHART*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BI_ELEMENT | VARCHAR2(30) | Foreign key to so_bi_element. | N |
| CHART_TYPE | VARCHAR2(15) | Foreign key to so_bi_chart_type table | N |
| DEFAULT_FLG | VARCHAR2(1) | Y/N indicator if the chart should be used as a default by the UI | N |
| COUNT_OR_PCT | VARCHAR2(10) | Metadata value that identies if the series of data used correspond to a count or a percentage. | N |
| BINNING_FLG | VARCHAR2(1) | Y/N indicator if binning should be used to draw the chart, | N |
| MULTIPLE_SERIES_FLG | VARCHAR2(1) | Y/N indicator if multiple series of data should be used to draw the chart. | N |
| DFLT_BINS | NUMBER(3) | This value is only relevant for charts that allow binning. This is the default number of bins that should be used by the chart. | Y |
| MIN_BINS | NUMBER(3) | This value is only relevant for charts that allow binning. This is the minimum number of bins that should be used by the chart. | Y |
| MAX_BINS | NUMBER(3) | This value is only relevant for charts that allow binning. This is the maximum number of bins that should be used by the chart. | Y |

**Notes:**

This is a lookup and metadata table that provides informations about the distinct options and parameters that should be used by the UI to draw charts. This object is used by development to configure BI modules.

## SO_BI_SECTION

This table contains the list of possible BI sections that are configured within the UI to use metadata and configuration values from the database.

*Table B–98 SO_BI_SECTION*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BI_SECTION | VARCHAR2(15) | Unique identifier for the BI Section. This value is static. The list of recognized values is provided by the application. The ID is used by the UI to link BI elements and objects to specific sections of the UI. | N |
| DESCR | VARCHAR2(30) | BI section description. | N |

**Notes:**

This is a lookup table that provides BI section information that can be supported by the application. It is referenced by other tables. The values in this table usually reference a section within the distinct UI trains stops.

## SO_DISPLAY_STYLE_FIXTURE_STG

This is the compatibility table between display styles and fixture types.

*Table B–99 SO_DISPLAY_STYLE_FIXTURE_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| DISPLAY_STYLE_KEY | VARCHAR2(80) | Position of the fixture on the Z axis. Origin point: bottom, left, back (within the bay). | Y |
| FIXTURE_TYPE | VARCHAR2(80) | Fixture type that supports the display style (Shelf, Pegboard, or Freezer chest). | Y |

**Notes:**

This is a compatibility table that defines which display styles can be used for the distinct fixture types.

## SO_DISP_STYLE_ORIENTATION_STG

This table provides a cross reference between display style and product valid orientations.

*Table B–100 SO_DISP_STYLE_ORIENTATION_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| DISPLAY_STYLE_KEY | VARCHAR2(80) | External display style ID. | Y |
| ORIENTATION_KEY | VARCHAR2(80) | External orientation ID. | Y |
| DEFAULT_FLG | VARCHAR2(30) | Y indicates the orientation should be consider as the default for the display style. N indicates the orientation is valid for the display style but not a default. | Y |

**Notes:**

This table contains the list of valid orientations for a given display style. The table includes a default orientation that should be used for each display style.

## SO_DISPLAY_STYLE_STG

This table contains the list of available display styles for products.

*Table B–101 SO_DISPLAY_STYLE_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| KEY | VARCHAR2(80) | External ID that identified the display style. | Y |
| NAME | VARCHAR2(80) | Display style name. | Y |
| DESCR | VARCHAR2(80) | Display style description. | Y |
| DEPTH | NUMBER(18,4) | Product depth relevant to default front-0 position. | Y |
| HEIGHT | NUMBER(18,4) | Product height relevant to default front-0 position. | Y |
| WIDTH | NUMBER(18,4) | Product width relevant to default front-0 position. | Y |
| FINGER_SPACE_ABOVE | NUMBER(18,4) | Product gap above between same product. | Y |
| FINGER_SPACE_BELOW | NUMBER(18,4) | Product gap beside between same product. | Y |
| FINGER_SPACE_BEHIND | NUMBER(18,4) | Product gap behind between same product. | Y |
| INTER_PRODUCT_GAP | NUMBER(18,4) | Gap between products. This field captures gap beside between different products. | Y |
| MAX_STACK | NUMBER(10) | Number of items that can be stacked together; equal to 1 if not stackable. | Y |
| NESTING_HEIGHT | NUMBER(18,4) | Product nesting height. The product does not allow nesting if all nesting dimensions are 0. | Y |
| NESTING_WIDTH | NUMBER(18,4) | Product nesting width. The product does not allow nesting if all nesting dimensions are 0. | Y |
| NESTING_DEPTH | NUMBER(18,4) | Product nesting depth. The product does not allow nesting if all nesting dimensions are 0. | Y |
| COLOR | VARCHAR2(30) | Product color. It can be NULL. | Y |
| DISPLAY_UNITS | NUMBER(3) | For unit display style it is 1; otherwise, it is >1. | Y |
| TYPE | VARCHAR2(80) | Display style type. CASE, UNIT, TRAY or other display style type. | Y |

**Notes:**

A display style defines the product physical dimensions as well as the different options that can be used to place the product within a planogram.

## SO_DISPLAY_STYLE_TYPE

This table gives the discrete unit of display for a given product that are set up in MSM. Examples of display style types are: single/unit, tray, case, and pallet.

*Table B–102 SO_DISPLAY_STYLE_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the display style type. This value is static. The list of recognized values is provided by the application. | N |
| DESCR | VARCHAR2(80) | Display style type description. | Y |

**Notes:**

This is a lookup table that provides display style types that can be supported by the application. It is referenced by other tables.

This table gives the discrete unit of display for a given product that are set up in MSM. Examples of display style types are: single/unit, tray, case, and pallet.

## SO_FIXTURE_DISP_CONFIG_STG

This table receives the historical planogram product data for shelf fixtures.

*Table B–103 SO_FIXTURE_DISP_CONFIG_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| DISPLAY_STYLE_KEY | VARCHAR2(80) | External ID that identifies a display style associated to a single product. | Y |
| BAY_KEY | VARCHAR2(80) | Bay external ID. Bay is a direct dependent of the planogram. | Y |
| FIXTURE_KEY | VARCHAR2(80) | Fixture external ID. Fixture is a direct dependent of the Bay. | Y |
| SHELF_KEY | VARCHAR2(80) | Shelf external ID. Shelf is a direct dependent of shelf fixture. | Y |
| ORIENTATION_KEY | VARCHAR2(80) | External ID that identifies orientation used. | Y |
| POS_X | NUMBER(18,4) | Position of the product on the X axis. Origin point: bottom, left, back. | Y |
| POS_Y | NUMBER(18,4) | Position of the product on the Y axis. Origin point: bottom, left, back. | Y |
| POS_Z | NUMBER(18,4) | Position of the product on the Y axis. Origin point: bottom, left, back. | Y |
| FACING_QUANTITY | NUMBER(5) | Number of facings of the product. | Y |

**Notes:**

This table contains the list of products and distribution of them across historical planograms. It includes the position and orientation of the products within the planogram as well as the number of facings per products. This data is used to estimate facing lift.

## SO_FIXTURE_STG

This table contains the list of fixtures within a historical planogram.

*Table B–104 SO_FIXTURE_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| KEY | VARCHAR2(80) | External fixture identifier. | Y |
| FIXTURE_TYPE | VARCHAR2(80) | Fixture type values can one of the following: Shelf, Pegboard or Freezer Chest. | Y |
| DEPTH | NUMBER(18,4) | Fixture depth. | Y |
| HEIGHT | NUMBER(18,4) | Fixture height. | Y |
| WIDTH | NUMBER(18,4) | Fixture width. | Y |
| VERTICAL_SPACING | NUMBER(18,4) | Vertical spacing. This value is only relevant for pegboard fixtures. | Y |
| HORIZONTAL_SPACING | NUMBER(18,4) | Horizontal spacing. This value is only relevant for pegboard fixtures. | Y |
| MAX_LENGTH | NUMBER(18,4) | Max length. This value is only relevant for pegboard fixtures. | Y |
| CAPACITY_X | NUMBER(18,4) | Freezer Chest Capacity X (length). This value is only relevant for freezer chest fixtures. | Y |
| CAPACITY_Y | NUMBER(18,4) | Freezer Chest Capacity Y (depth). This value is only relevant for freezer chest fixtures. | Y |
| CAPACITY_Z | NUMBER(18,4) | Freezer Chest Capacity Z (height). This value is only relevant for freezer chest fixtures. | Y |

**Notes:**

This table contains the list of fixtures that define the historical planogram layout. Fixtures are planogram components within a Bay; each bay can include one or more fixtures.

## SO_FIXTURE_TYPE

This table defines what kind of fixtures are allowed in a POG: shelf, pegboard, or freezer chest.

*Table B–105 SO_FIXTURE_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the fixture type. This value is static. The list of recognized values is provided by the application. | N |
| DESCR | VARCHAR2(80) | Fixture type description. | Y |

**Notes:**

This is a lookup table that provides fixture types supported by the application. It is referenced by other tables.

The supported fixture types are: Shelf, Pegboard and Freezer Chest.

## SO_FRONT_FACING

This table contains the list of possible products sides that can be used as front facings supported by the application.

*Table B–106 SO_FRONT_FACING*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the front facing side. This value is static. The list of recognized values is provided by the application. | Y |
| DESCR | VARCHAR2(80) | Front facing description. | Y |

**Notes:**

This is a lookup table that provides the list of product sides that can be used as the front facings supported by the application. It is referenced by other tables.

Front facing values can be: front, back, top, bottom, left, and right.

## SO_GLOBAL_VALDT_STG

This table is an auxiliary table that is used within the global validation process. This table is used to meet the loader framework that requires a staging table. The loaders framework is used by the global validation process to perform validations (only).

*Table B–107 SO_GLOBAL_VALDT_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| VALDT_TBL_NAME | VARCHAR2(30) | Auxiliary column to host a table name | Y |

## SO_INT_TRANSFORMATION_KEY

This table is used to help perform interface data transformation. The values in this table are used to align data from external sources with the data expected by ASO. It helps to isolate translation conversion issues.

*Table B–108 SO_INT_TRANSFORMATION_KEY*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| SO_INTERFACE | VARCHAR2(30) | This string identifies the interface that makes use of the values. That is, ASO-MSM (indicates the interface that sends data from ASO to MSM). | N |
| SO_DATA_ELEMENT | VARCHAR2(30) | This is the reference/name of the ASO column that is used on the numeric or text ASO value fields. | N |
| SO_NUM_DATA_VALUE | NUMBER(18) | This field is used if the ASO value is numeric. If the value is not numeric, this field will be set to -1. | N |
| SO_TEXT_DATA_VALUE | VARCHAR2(80) | This field will be used if the ASO value is text. The field will be set to X (instead of NULL) if the SO value is numeric. | N |
| INT_NUM_DATA_VALUE | NUMBER(18) | This field will be used if the interface value is transformed to a numeric value. | Y |
| INT_TEXT_DATA_VALUE | VARCHAR2(80) | This field will be used if the interface value is transformed to a text value. | Y |

## SO_LOC_OPTIMIZATION_LEVEL

This table contains the distinct list of possible location optimization levels.

*Table B–109 SO_LOC_OPTIMIZATION_LEVEL*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for location optimization level. This value is static. The list of recognized values is provided by the application. | N |
| NAME | VARCHAR2(80) | Location optimization level name. This value should not be used for display within the UI; it is used by the backend. | Y |
| DESCR | VARCHAR2(200) | Location Optimization level description. | Y |

**Notes:**

This is a lookup table that provides location optimization levels that can be supported by the application. It is referenced by other tables. Values include Cluster and Store.

This is the level at which the optimization is performed. In some cases it is acceptable to perform optimization at a higher level (cluster). This reduces processing time in exchange for using data aggregated at the cluster level.

## SO_ORIENTATION

This table describes how a products package faces the front of the fixture.

The orientation has two pieces of information: what part of the package is to the front (front, back, top, bottom, left, right) and how it is rotated in degrees (0, 90, 180, 270), for a total of 24 possible orientations. How a product is oriented determines how much space it takes in the x,y,z directions on a fixture.

Legal orientations are all orientations that can be used for the product.

The default orientation is the legal orientation that is normally used for the product.

*Table B–110 SO_ORIENTATION*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the run strategy. This value is static. The list of recognized values is provided by the application. Each one of this orientation IDs is already identified and recognized to correspond to a specific front facing and rotation. The ID values should not be changed. | N |
| ORIENTATION_EXT_KEY | VARCHAR2(80) | Orientation external ID. | Y |
| ROTATION_DEGREES | NUMBER(3) | Rotation in degrees used in the specific rotation. Valid values are 0, 90, 180, and 270. | Y |
| FRONT_FACING_ID | NUMBER(10) | This is a foreign key to so_front_facing table. It indicates the product side that should be used as a front facing. | N |

**Notes:**

This is a lookup table that provides orientations that can be supported by the application. It is referenced by other tables.

It describes how a product’s package faces the front of the fixture.

The orientation has two pieces of information: the part of the package that goes to the front (front, back, top, bottom, left, right) and how it is rotated in degrees (0, 90, 180, 270), for a total of 24 possible orientations. How a product is oriented determines how much space it takes in the x,y,z directions on a fixture. Legal orientations are all orientations that can be used for the product.

The default orientation is the legal orientation that is normally used for the product.

## SO_ORIENTATION_STG

This table contains the list of available orientations that can be used within planograms.

***Table B–111 SO_ORIENTATION_STG***

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| KEY | VARCHAR2(80) | External identifier for the orientation. | Y |
| FRONT_FACING | VARCHAR2(80) | The front facing can be: front, back, top, bottom, left, right. | Y |
| ROTATION_DEGREE | NUMBER(3) | The rotation degree can be 0, 90, 180, 270. | Y |

**Notes:**

This table describes how a product package faces the front of the fixture. It has a total of 24 possible orientations.

## SO_PEGBOARD_DISP_CONFIG_STG

This table contains the historical planogram product data for pegboard and freezer chest fixtures.

***Table B–112 SO_PEGBOARD_DISP_CONFIG_STG***

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| DISPLAY_STYLE_KEY | VARCHAR2(80) | External ID that identifies a display style associated to a single product. | Y |
| BAY_KEY | VARCHAR2(80) | Bay external ID. Bay is a direct dependent of the planogram. | Y |
| FIXTURE_KEY | VARCHAR2(80) | Fixture external ID. Fixture is a direct dependent of the Bay. | Y |
| ORIENTATION_KEY | VARCHAR2(80) | External ID that identifies orientation used. | Y |
| POS_X | NUMBER(18,4) | Position of the product on the X axis. Origin point: bottom, left, back. | Y |
| POS_Y | NUMBER(18,4) | Position of the product on the Y axis. Origin point: bottom, left, back. | Y |
| POS_Z | NUMBER(18,4) | Position of the product on the Z axis. Origin point: bottom, left, back. | Y |
| FACING_QUANTITY | NUMBER(5) | Number of facings of the product. | Y |

**Notes:**

This table contains the list of products and distribution of them across historical planograms. It includes the position and orientation of the products within the planogram as well as the number of facings per products This data is used to estimate facing lift.

## SO_POG_ASSORT_MAPPING_STG

This staging table contains the cross reference data to perform POG to assortment mapping.

*Table B–113 SO_POG_ASSORT_MAPPING_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POG_DEPT_KEY | VARCHAR2(80) | This is the POG dept key. This is a POG hierarchy external key known to the external source. It is a mandatory value. | Y |
| POG_CATEGORY_KEY | VARCHAR2(80) | This is the POG category key. This is a POG hierarchy external key known to the external source. It is a mandatory value. | Y |
| POG_SUB_CATEGORY_KEY | VARCHAR2(80) | This is the POG subcategory key. This is a POG hierarchy external key known to the external source. It is a mandatory value. | Y |
| ASSORT_PRODUCT_LEVEL | VARCHAR2(80) | This is an identifier to the product level within the product hierarchy. This value must match the product hierarchy available within SO. | Y |
| ASSORT_PRODUCT_KEY | VARCHAR2(80) | This is an identifier to a node within the merchandise hierarchy. It could be a specific product or any other node not higher than the assortment product category level within the merchandise hierarchy. | Y |
| DEMAND_SPREAD_FACTOR | NUMBER(6,3) | This is the demand spread factor. This value is normally null, meaning a 100% demand is assigned to the POG node. In specific cases where the product is placed on multiple POG nodes, a demand spread factor can be used to split the demand across those multiple POGs. | Y |

**Notes:**

This table contains the POG hierarchy to assortment product mapping information. This data is used to identify that POG should be used for each product within an assortment.

## SO_POG_ASSORT_SEAS_MAPPING_STG

This staging table contains the cross reference data for the assortment-to-POG season mapping.

*Table B–114 SO_POG_ASSORT_SEAS_MAPPING_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POG_DEPT_KEY | VARCHAR2(80) | This is the POG dept key. This is a POG hierarchy external key known to the external source. It is a mandatory value. | Y |
| POG_CATEGORY_KEY | VARCHAR2(80) | This is the POG category key. This is a POG hierarchy external key known to the external source. It is a mandatory value. | Y |
| POB_SUB_CATEGORY_KEY | VARCHAR2(80) | This is the POG subcategory key. This is a POG hierarchy external key known to the external source. It is a mandatory value. | Y |

*Table B–114 (Cont.) SO_POG_ASSORT_SEAS_MAPPING_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| SEASONAL_ATTRIBUTE | VARCHAR2(30) | This field refers to a specific year independent time period (season) for a Category Management assortment and a POG set. Examples include Spring, holiday, back to school, year-round, Fall, and Winter. | Y |
| MIN_ASSORT_START_DB | DATE | The year component is irrelevant; it should be 0000. This is a year independent time period. The assortment start date is matched within the date range specified by the min assort start date and the max assort start date. | Y |
| MAX_ASSORT_START_DT | DATE | The year component is irrelevant; it should be 0000. This is a year independent time period. The assortment start date is matched within the date range specified by the min assort start date and the max assort start date. | Y |

**Notes:**

Once the mapping from product to POG has been performed, a second pass examines the data to identify the specific POG season to use, based on the assortment start date.

## SO_POG_BAY_STG

This table contains the list of bays that are used to build a planogram.

*Table B–115 SO_POG_BAY_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| BAY_KEY | VARCHAR2(80) | Bay external ID. Bay is a direct dependent of the planogram. | Y |
| POG_KEY | VARCHAR2(80) | External planogram identifier. | Y |
| BAY_SEQUENCE | NUMBER(3) | Sequence from left to right in which the bay appear within the planogram. | Y |

**Notes:**

A bay is the level under the planogram that is used to position fixtures to build the final planogram layout. The bay is directly linked to a unique planogram; fixtures are then linked to the bay.

## SO_POG_DISPLAY_STYLE_STG

This table is the cross reference between historical planograms and product display styles.

*Table B–116 SO_POG_DISPLAY_STYLE_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POG_KEY | VARCHAR2(80) | External planogram identifier. | Y |
| DISPLAY_STYLE_KEY | VARCHAR2(80) | External display style identifier. This identifier links a historical planogram with a specific product. | Y |

**Notes:**

This table contains the list of products (based on the display style) that use this historical planogram.

## SO_POG_FIXT_CONFIG_ALGORITHM

This table keeps the list of available algorithms to perform the shelf fixture smart start process (create shelves for empty fixtures).

*Table B–117 SO_POG_FIXT_CONFIG_ALGORITHM*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Parameter unique identifier. | N |
| NAME | VARCHAR2(80) | Parameter name. | N |
| DESCR | VARCHAR2(200) | Parameter description. | Y |
| DEFAULT_FLG | VARCHAR2(1) | This flag is set to 'Y' for the default algorithm. The default algorithm is used during the optimization location generation before the user has the opportunity to pick or change parameters. | N |
| ENABLED_FLG | VARCHAR2(1) | This flag is used to indicate if the algorithm is available or active. | N |

## SO_POG_FIXTCONF_ALG_PARAM

This table is used to store the list of different algorithm parameters the user can customize while running the fixture smart start process.

*Table B–118 SO_POG_FIXTCONF_ALG_PARAM*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | VARCHAR2(20) | Parameter unique identifier. | N |
| POG_FIXT_CONFIG_ ALGORITHM_ID | NUMBER(10) | Foreign key linking the parameter to a specific algorithm. | N |
| NAME | VARCHAR2(80) | Parameter name. | N |
| DESCR | VARCHAR2(200) | Parameter description. | Y |
| DFLT_VALUE | NUMBER(18,4) | Default parameter value that the UI suggests to the user. | N |
| MIN_VALUE | NUMBER(18,4) | Minimum acceptable parameter value. This value is used to validate the user entries. | N |
| MAX_VALUE | NUMBER(18,4) | Maximum acceptable parameter value. This value is used to validate the user entries. | N |
| PARAM_SEQ | NUMBER(18,4) | Sequence in which the parameter should be presented or retrieved to the user within the UI. | N |

## SO_POG_FIXTURE_TYPE

This entity is used to identify distinct POG types (Partial Shelf Fixture Planogram (PSFP) or Complete Planogram (CFP)).

*Table B–119 SO_POG_FIXTURE_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POG_FIXTURE_TYPE | VARCHAR2(10) | Two types: Partial Shelf Fixture Planogram (PSFP) or Complete Fixture Planogram (CFP) | N |
| DESCR | VARCHAR2(80) | Fixture type description. | Y |
| UI_DESCR | VARCHAR2(80) | Fixture type description, customized for display by the UI. | Y |

## SO_POG_SOURCE

This table tracks the different POG sources (for example, MSM, Promotional POG from external source, and Internal POG from ASO).

*Table B–120 SO_POG_SOURCE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the planogram source. This value is static. The list of recognized values is provided by the application. More sources can be added to this table; each must have a unique identifier. | N |
| DESCR | VARCHAR2(80) | Planogram source description. | Y |

## SO_POG_STATUS

Table that contains the list of possible planogram status that can be supported by the application.

*Table B–121 SO_POG_STATUS*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the planogram status. This value is static. The list of recognized values is provided by the application. | N |
| DESCR | VARCHAR2(80) | Planogram status description. Values include approved, rejected, pending, and received. | Y |
| POG_SET_FLG | VARCHAR2(1) | Indicates is the status applies to the POG set instead of the POG. | N |

**Notes:**

This is a lookup table that provides planogram status supported by the application. it is referenced by other tables.

## SO_POG_STG

This table contains planogram level details for the planogram header data.

*Table B–122 SO_POG_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| KEY | VARCHAR2(80) | External planogram identifier. | Y |
| NAME | VARCHAR2(80) | Planogram name. | Y |
| DESCR | VARCHAR2(80) | planogram description. | Y |

*Table B–122 (Cont.) SO_POG_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| SEASON_CODE | VARCHAR2(30) | Seasonal code used by the historical planogram. | Y |
| SEASONAL_ATTRIBUTE | VARCHAR2(30) | Seasonal attribute used for the historical planogram. | Y |
| EFFECTIVE_START_DT | DATE | Planogram start date. | Y |
| EFFECTIVE_END_DT | DATE | Planogram end date. | Y |
| STATUS | VARCHAR2(30) | Identifier that describes the planogram status. | Y |
| CATEGORY_KEY | VARCHAR2(80) | POG category key. The second lowest level of POG category hierarchy. | Y |
| CATEGORY_NAME | VARCHAR2(80) | POG category name. | Y |
| SUB_CATEGORY_KEY | VARCHAR2(80) | POG sub-category key. The lowest level of POG category hierarchy. | Y |
| SUB_CATEGORY_NAME | VARCHAR2(80) | POG sub-category name. | Y |
| DEPT_KEY | VARCHAR2(80) | POG department key. | Y |
| DEPT_NAME | VARCHAR2(80) | POG department name. | Y |
| LENGTH | NUMBER(18,4) | The total length of a planogram. It must be equal to the sum of the length for all the bays within the planogram. | Y |
| DEPTH | NUMBER(18,4) | The total depth of a planogram. It must be equal to the greatest depth within all the fixtures in the planogram. | Y |
| HEIGHT | NUMBER(18,4) | The total height of a planogram. It must be equal to the highest fixture within the planogram. | Y |

**Notes:**

The data in this table is used internally to generate the distinct POG set. The content of this table is transformed into planograms and POG sets. The rows within this table correspond to historical planograms. The layouts are received from external sources.

## SO_POG_STORE_STG

This table contains the list of stores that used the historical planogram.

*Table B–123 SO_POG_STORE_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POG_KEY | VARCHAR2(80) | External planogram identifier. | Y |
| STORE_KEY | VARCHAR2(80) | This is the external store ID, known and shared across applications. | Y |
| EFFECTIVE_START_DT | DATE | Start date for which the historical planogram is valid for the given store. | Y |
| EFFECTIVE_END_DT | DATE | End date for which the historical planogram is valid for the given store. | Y |

**Notes:**

This table provides a cross reference between historical planograms and stores for which the planogram is valid (depending on dates).

## SO_POG_STORE_CDA_STG

This is a staging table to load customer defined attributes for POG/store combinations. These attributes are static values that are used as informational attributes within the UI.

*Table B–124 SO_POG_STORE_CDA_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POG_KEY | VARCHAR2(80) | External planogram identifier | Y |
| STORE_KEY | VARCHAR2(80) | This is the external store ID, known and shared across applications. | Y |
| ATTR_NUM_VALUE_1 | NUMBER(18,4) | Generic numeric value | Y |
| ATTR_NUM_VALUE_2 | NUMBER(18,4) | Generic numeric value | Y |
| ATTR_NUM_VALUE_3 | NUMBER(18,4) | Generic numeric value | Y |
| ATTR_NUM_VALUE_4 | NUMBER(18,4) | Generic numeric value | Y |
| ATTR_NUM_VALUE_5 | NUMBER(18,4) | Generic numeric value | Y |
| ATTR_NUM_VALUE_6 | NUMBER(18,4) | Generic numeric value | Y |
| ATTR_NUM_VALUE_7 | NUMBER(18,4) | Generic numeric value | Y |
| ATTR_NUM_VALUE_8 | NUMBER(18,4) | Generic numeric value | Y |
| ATTR_NUM_VALUE_9 | NUMBER(18,4) | Generic numeric value | Y |
| ATTR_NUM_VALUE_10 | NUMBER(18,4) | Generic numeric value | Y |
| ATTR_DATE_VALUE_1 | DATE | Generic date value | Y |
| ATTR_DATE_VALUE_2 | DATE | Generic date value | Y |
| ATTR_DATE_VALUE_3 | DATE | Generic date value | Y |
| ATTR_DATE_VALUE_4 | DATE | Generic date value | Y |
| ATTR_DATE_VALUE_5 | DATE | Generic date value | Y |
| ATTR_STRING_VALUE_1 | VARCHAR2(80) | Generic string value | Y |
| ATTR_STRING_VALUE_2 | VARCHAR2(80) | Generic string value | Y |
| ATTR_STRING_VALUE_3 | VARCHAR2(80) | Generic string value | Y |
| ATTR_STRING_VALUE_4 | VARCHAR2(80) | Generic string value | Y |
| ATTR_STRING_VALUE_5 | VARCHAR2(80) | Generic string value | Y |
| ATTR_PCT_VALUE_1 | NUMBER(5,4) | Generic percentage value | Y |
| ATTR_PCT_VALUE_2 | NUMBER(5,4) | Generic percentage value | Y |
| ATTR_PCT_VALUE_3 | NUMBER(5,4) | Generic percentage value | Y |
| ATTR_PCT_VALUE_4 | NUMBER(5,4) | Generic percentage value | Y |
| ATTR_PCT_VALUE_5 | NUMBER(5,4) | Generic percentage value | Y |

## SO_POGSET_ASSORT_ASSIGN_TYPE

This table contains the distinct mapping types that can be used between assortments and planograms.

*Table B–125 SO_POGSET_ASSORT_ASSIGN_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| POGSET_ASSORT_ASSIGN_TYPE | VARCHAR2(10) | Unique identifier for the mapping assignation type. This value is static. The list of recognized values is provided by the application. | N |
| DESCR | VARCHAR2(200) | Mapping assignation type description. | Y |

**Notes:**

This table contains assortment to planogram assignation types. It is referenced by other tables. Values include automatic, escalation, manual, current, and unassigned.

## SO_PROD_CONSTR_RANGE_VALUES

This table contains the list of product constraint values supported by the application.

*Table B–126 SO_PROD_CONSTR_RANGE_VALUES*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the constraint value. This value is static. The list of recognized values is provided by the application. | N |
| PRODUCT_CONSTRAINT_TYPE | VARCHAR2(10) | Foreign key from so_product_constraint_type table. It identifies the product constraint for which the value can be used. | N |
| VALUE | VARCHAR2(30) | The actual value to be presented by the UI. | Y |
| MIN_FLG | VARCHAR2(1) | Y indicates it can be used as minimum value. | Y |
| MAX_FLG | VARCHAR2(1) | Y indicates it can be used as maximum value. | Y |
| UNIT_FLG | VARCHAR2(1) | Y indicates it can be used as a Units option. | Y |
| UI_DFLT_FLG | VARCHAR2(1) | Y indicates this is the default UI value. | Y |
| DB_DFLT_FLG | VARCHAR2(1) | Y indicates this is the default value to be used by the DB processes. | Y |
| VALUE_ORDER | NUMBER(10) | This field is used to sort the data in the appropriate way for displaying, since strings and numbers are mixed in the same table. | N |

**Notes:**

This table contains the range and a list of values that are supported by the application. It is referenced by other tables and is used to populate a constraint list of values and options presented to the user to configure run product constraints.

## SO_PROD_DISPLAY_STYLE_STG

This table contains display style cross references.

*Table B–127 SO_PROD_DISPLAY_STYLE_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PRODUCT_KEY | VARCHAR2(80) | This is the external ID that is known and shared across applications. This cannot be a placeholder product. | Y |
| DISPLAY_STYLE_KEY | VARCHAR2(80) | External ID that identifies the display style. | Y |
| DEFAULT_FLG | VARCHAR2(1) | Y indicates the default display style for a given product. N indicates the combination should not be considered as a default. Each product should have one default display style. | Y |

**Notes:**

This table contains product to display style mapping. It provides a list of display styles available to choose for certain product.

## SO_PROD_LOC_REPL_PARAM_STG

This table contains the replenishment data corresponding to product/store combinations.

*Table B–128 SO_PROD_LOC_REPL_PARAM_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PRODUCT_KEY | VARCHAR2(1) | This is the external ID that is known and shared across applications. For placeholder products, this field contain the Category Management placeholder product key. | Y |
| LOCATION_KEY | VARCHAR2(80) | This is the external store ID, known and shared across applications. | Y |
| CASEPACK | VARCHAR2(80) | Product casepack for the given store. | Y |
| REPLENISHMENT_FREQ | NUMBER(18,4) | Replenishment frequency (RF) = number of replenishments to the shelf per week. | Y |
| REPLENISHMENT_TYPE | NUMBER(18,4) | Replenishment source/type has two options: 1 = from DC/vendor and 2 = from back room. | Y |
| TRANSIT_TIME | NUMBER(10,2) | Transit time (TT) is the number of days it takes an order to go from the source (DC or back room) to the shelf. | Y |
| SHELF_REPLENISHMENT_TT | NUMBER(10) | Shelf replenishment trigger type has three options: 1 = cover demand over replenishment period + transit time, 2 = replenishment when inventory gets to a target percent of capacity, and 3 = replenishment when a casepack can fit. | Y |
| SHELF_REPLENISHMENT_PARAM | NUMBER(18,4) | Shelf replenishment parameter (currently only applies for option 2). | Y |
| STDEV_BOOSTER | NUMBER(10,6) | Standard deviation booster (number greater than or equal to 0, makes sense to limit to 1). | Y |
| DAYS_OF_SALES_PER_WK | NUMBER(3,2) | Days of sales per week (number between 1 and 7). | Y |
| FACINGS_LIFT | NUMBER(5,4) | Facing lift parameter. | Y |

**Notes:**

Replenishment parameters are not directly linked to any assortment. This is generic data; however, these parameters must exist for all the known product/store combinations provided within an assortment.

## SO_PROD_STACK_HEIGHT_LIMIT_STG

This table is used to accept an optional client feed that provides product-specific stacking height limits.

*Table B–129 SO_PROD_STACK_HEIGHT_LIMIT_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PRODUCT_KEY | VARCHAR2(80) | This is the external ID that is known and shared across applications. | N |
| STACK_HEIGHT_LIMIT | NUMBER(18,4) | This is the stacking height limit for the specific product. The value here must be provided using the same units of measure used for all other product dimensions. | N |
| ENABLED_FLG | VARCHAR2(1) | This flag indicates if the product stacking height limit should be used or not. Y means the value specified here will be used; N means the value will be ignored and the application global value will be used instead for the product. | N |

**Notes:**

Replenishment parameters are not directly linked to any assortment. This is generic data; however, these parameters must exist for all the known product/store combinations provided within an assortment.

## SO_PRODUCT_CONSTRAINT_TYPE

This table contains the list of product constraint types supported by the application.

*Table B–130 SO_PRODUCT_CONSTRAINT_TYPE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| PRODUCT_CONSTRAINT_TYPE | VARCHAR2(10) | Unique identifier for the product constraint type. This value is static. The list of recognized values is provided by the application. | N |
| NAME | VARCHAR2(50) | Product constraint type name. This value is used by the UI. | Y |
| PROD_GROUP_FLG | VARCHAR2(1) | A value of Y identifies the rows that are valid Product Group Constraints. | Y |

**Notes:**

This lookup table contains product constraint types supported by the application. It is referenced by other tables.

Valid product constraints include IN (Inclusion), CR (Capacity Range), CRU (Capacity Range Units), FR (Facing Range), HR (Height Range). Valid group product constraints include MF (Match facing), SS (Same Shelf), CF (Choose From), AN (All or nothing).

## SO_PRODUCT_PRIORITY

This table contains the list of possible product priority values supported.

*Table B–131 SO_PRODUCT_PRIORITY*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the product priority. This value is static. The list of recognized product priorities is provided by the application. | N |
| DESCR | VARCHAR2(80) | Product Priority description. | Y |

**Notes:**

This table contains the assortment status. It is referenced by other tables. Values include 1=Mandatory, 2=Core, 3=Optional, -1=Dropped.

## SO_REPL_PARAM_DESCRIPTION

This table keeps the list of replenishment parameters the user can change. These parameters have a defined list of valid values that are kept in this table so they can be used by the UI to present them to the user.

*Table B–132 SO_REPL_PARAM_DESCRIPTION*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | VARCHAR2(10) | Unique replenishment parameter identifier. | N |
| REPL_PARAMETER_NAME | VARCHAR2(30) | Replenishment parameter name. This is the higher level parameter. | N |
| DESCR | VARCHAR2(80) | This is the description corresponding to the specific value the user can use for the parameter. | Y |
| UI_DESCR | VARCHAR2(80) | This is a short description to be presented to the user, that is,. labels on a list of values. | Y |
| VALUE_ORDER | NUMBER95) | This is the order in which the parameter value should be presented by the UI. | Y |

## SO_RUN_CLUSTER_SRC

This table contains the distinct mapping run cluster sources, including whether the cluster was copied from the assortment, generated by the application, loaded, or manually created.

*Table B–133 SO_RUN_CLUSTER_SRC*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the run cluster source. This value is static. The list of recognized values is provided by the application. | N |
| NAME | VARCHAR2(80) | Run cluster source. This value must not be used for display within the UI; it is used by the backend. | Y |
| DESCR | VARCHAR2(200) | Run cluster source description. | Y |

**Notes:**

This information is not relevant for the end user; it for internal use only.

## SO_RUN_OBJECTIVE

This table contains the list of possible run objectives that are supported by the application.

*Table B–134 SO_RUN_OBJECTIVE*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the run objective. This value is static. The list of recognized values is provided by the application. | N |
| NAME | VARCHAR2(80) | Run objective name. This value must not be used for display within the UI; it is used by the backend. | Y |
| DESCR | VARCHAR2(200) | Run objective description. | Y |

**Notes:**

This table contains run objectives supported by the application.It is referenced by other tables. Values include Run for Assortment Rationalization, Run for Promo POG Opt, and Refresh AR run.

## SO_RUN_OBJECTIVE_FUNC

The user has eight options for the objective function:

- Maximize sales value - un-weighted/weighted by IPI
- Maximize margin value - un-weighted/weighted by IPI
- Maximize sales units - un-weighted/weighted by IPI
- Maximize total sales value/average OH value - un-weighted/weighted by IPI

*Table B–135 SO_RUN_OBJECTIVE_FUNC*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the run objective function. This value is static. The list of recognized values is provided by the application. | N |
| NAME | VARCHAR2(80) | Run objective function name. This value must not be used for display within the UI; it is used by the backend. | Y |
| DESCR | VARCHAR2(200) | Run objective function description. | Y |

**Notes:**

This table contains run objective functions that are supported by the application. It is referenced by other tables.

## SO_RUN_PRODUCT_DROP_REASON

This table contains the list of possible reasons for which a product can be dropped from the optimization process.

*Table B–136 SO_RUN_PRODUCT_DROP_REASON*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| RUN_PRODUCT_DROP_ REASON | VARCHAR2(20) | Unique identifier for the drop product reason. This value is static. The list of recognized values is provided by the application. | N |
| DESCR | VARCHAR2(50) | Description of the reason that caused the product to be dropped from optimization. | Y |

**Notes:**

Values for the reason include NS (No Solution), SC (Solver Choice), UC (User Constraints), and MO (Manual Override).

## SO_RUN_STATUS

This table contains a list of possible run status, including Not Started, Warning, Processing, and OK.

*Table B–137 SO_RUN_STATUS*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the run status. This value is static. The list of recognized run statuses is provided by the application. | N |
| DESCR | VARCHAR2(80) | Run status description. | Y |
| UI_NAME | VARCHAR2(50) | Name used within the UI to recognize the status. | N |
| UI_DESCR | VARCHAR2(80) | Status description displayed by the UI. | Y |

**Notes:**

This table contains the values for the run status that are used by the application.It is referenced by other tables. Values include Initialized, Not Started, Build Optimization Locations, Processing, Approved, and Exported.

## SO_RUN_STRATEGY

This table contains the distinct list of possible optimization strategies, including No Macro Space changes and Macro Space changes.

*Table B–138 SO_RUN_STRATEGY*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the run strategy. This value is static. The list of recognized values is provided by the application. | N |
| NAME | VARCHAR2(80) | Run strategy name. This value must not be used for display within the UI; it is used by the backend. | Y |
| DESCR | VARCHAR2(200) | Run strategy description. | Y |

**Notes:**

This table contains run optimization strategies that are supported by the application.

## SO_SHELF_STG

This table contains the list of shelves within a historical planogram that uses shelf fixtures.

*Table B–139 SO_SHELF_STG*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| KEY | VARCHAR2(80) | External shelf identifier. | Y |
| DEPTH | NUMBER(18,4) | Shelf depth. | Y |
| HEIGHT | NUMBER(18,4) | Shelf Height. This is the physical shelf height/thickness. | Y |
| WIDTH | NUMBER(18,4) | Shelf width. | Y |

**Notes:**

This table contains the list of shelves that define a shelf fixture within the historical planogram layout. Shelf are planogram components within a Shelf fixture; each shelf fixture can contain one or more shelves.

## SO_TASK

This table contains the list of run tasks.

*Table B–140 SO_TASK*

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the run task. This value is static. The list of recognized values is provided by the application. | N |
| NAME | VARCHAR2(80) | Name of the task. | Y |
| DESCR | VARCHAR2(200) | Run task description. | Y |
| TASK_SEQUENCE | NUMBER(3) | Sequence in which the task should be executed. | Y |
| CURRENT_FLG | VARCHAR2(1) | (Y/N) Flag that indicates if the task is current or not. If the task is not current then it should not be used. A task could become current at a later time. | Y |
| DELETE_FLG | VARCHAR2(1) | (Y/N) Flag that indicates the task has been deleted and should no longer be used. | Y |

**Notes:**

This is a lookup/operations table that keeps a list of tasks that need to be performed for a run. (Not currently in use).

## SO_TASK_STATUS

This table contains a list of possible task status, including Not Started, Warning, Processing, completed, and paused.

***Table B–141 SO_TASK_STATUS***

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | Unique identifier for the task status . This value is static. The list of recognized values is provided by the application. | N |
| DESCR | VARCHAR2(80) | Run task status description. | Y |

**Notes:**

This table contains the run task status that can be supported by the application. It is referenced by other tables.

## SO_UI_MODULE

This table contains the list of UI modules that make use of the metadata configuration saved within the database.

***Table B–142 SO_UI_MODULE***

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| UI_MODULE | VARCHAR2(30) | Unique identifier for the UI module. This value is static. The list of recognized values is provided by the application. | N |
| DESCR | VARCHAR2(50) | UI module description, usually matching the train stop name. | N |

**Notes:**

This is a lookup/metadata table that tracks the different UI modules that require some metadata from the database. It is referenced by other tables. These table contain entries for the different train stops and UI sub-modules.

## STAGE

This table defines processing stages for an application workflow.

***Table B–143 STAGE***

| Column Name | Data Type | Comments | Nullable? |
|---|---|---|---|
| ID | NUMBER(10) | PK for this table. | N |
| NAME | VARCHAR2(80) | The name for this stage. | Y |
| DESCR | VARCHAR2(255) | More descriptive text for this stage. | Y |
| SEQ | NUMBER(10) | The sequence in which this stage is executed. The numbers are sequential starting at 1, with no overlap. | Y |
| SRVC_TYPE | VARCHAR2(30) | The name of the database service type used for any stage processes. | N |
| SRVC_NAME | VARCHAR2(30) | The name of the database service that runs for this stage of processing. | N |
| APPL_CODE | VARCHAR2(20) | The application code that uses this set of stages. | Y |

**Notes:**

The ID in this table is a fixed ID and is referenced by application code. The NAME is also referenced by application code. Therefore, they cannot be changed.

The description column (DESCR) can be changed if desired.

The SEQ column denotes the sequence of the stages. The sequence of stages is generally not considered changeable, as the applications expect processes to run in a particular order. However, it is present to allow for the introduction of new stages between the existing stages.

The SRVC_TYPE is a fixed value to distinguish one application's stages from another application's stages. This value is fixed and is referenced by the application code and therefore cannot be changed. It can be a reference to a value in the RSE_SRVC_TYPE table.

The SRVC_NAME is a reference to the RSE_SRVC_CONFIG's NAME values. It enables system-generated code to exist, so that it is possible to dynamically invoke the appropriate database service implementation for a given stage.

The APPL_CODE is a application identifier to indicate which application the stages are used by. This value should be similar to the values used in the RSE_CONFIG's APPL_CODE column.

# C

# Log File Locations

A number of log file types are available for diagnostics in the following categories:

- Weblogic logs
- Data and script logs
- ASO Solver logs and dump files

## Weblogic Logs

Server logs are available for each administrative or managed server, at the following path:

<WLS DOMAIN>/servers/<SERVER NAME>/logs

Some common logs that can contain information relevant to the ORASE applications include:

- <SERVER NAME>.log
- <SERVER NAME>-diagnostic.log
- access.log

Refer to the Oracle WebLogic Support documentation for details on contents of these log files.

## Database Logs

Linux script logs are located in these subdirectories of the RSE HOME directory:

- ./common/data/log
- ./common/scripts/log
- ./cdm/cis/scripts/log
- ./cdm/cis/data/log
- ./cdm/mba/data/log
- ./cdm/mba/scripts/log
- ./cdm/dt/data/log
- ./cdm/dt/scripts/log
- ./cdm/cdt/data/log
- ./cdm/cdt/scripts/log
- ./so/data/log

- ./so/scripts/log

The data/log directories will contain logs for any SQL Loader file loading, while the scripts/log directories would contain logs for any .ksh scripts that were run.

Additionally, database logs can be found in the ORASE schema in a table called RSE_LOG_MSG. The database code has an option to log to this table. Any PL/SQL code errors will automatically have their error messages written to this table with a LOG_LEVEL='ERROR'.

It is also possible to enable the database PL/SQL code to write additional levels of messages to this table. The PL/SQL code supports conditional compilation options for the following:

- DEBUG_DATA - This compilation option is not a logging option, but it will make the code retain any transient tables, instead of allowing the code to purge them when the code is complete with the table. This is useful for debugging purposes, but can result in a large amount of used space, and therefore should be used sparingly.
- DEBUG_MODE - This compilation option will trigger logging of additional log messages so the code can report more granular information about the code is doing. This is not necessarily available for all routines, but setting this will not be detrimental. These log messages normally will appear with a LOG_LEVEL='DEBUG'.
- DEBUG_SQL - This compilation option will result in the PL/SQL code to record all dynamic SQL statements to be written to the log table. This can be useful for debugging. These log messages should appear as LOG_LEVEL= 'SQLSTMT'.
- DEBUG_TRACE - This compilation option will result in all procedures, and many functions to record start and end messages, along with a record of any parameters passed to the routines. These log messages should appear with a LOG_LEVEL=' TRACE'. These logging entries can be useful for tracking application performance.

An example to disable all the extra logging information is as follows:

```
-alter TYPE RSE_SLS_TXN_ETL_T compile PLSQL_CCFLAGS = 'DEBUG_DATA:FALSE, DEBUG_
MODE:FALSE, DEBUG_SQL:FALSE, DEBUG_TRACE:FALSE' ;
```

An example to enable all the optional debugging logging and data retention is as follows:

```
-alter TYPE RSE_SLS_TXN_ETL_T compile PLSQL_CCFLAGS = 'DEBUG_DATA:TRUE, DEBUG_
MODE:TRUE, DEBUG_SQL:TRUE, DEBUG_TRACE:TRUE' ;
```

Any combination of options may be set to TRUE or FALSE as required.

### ASO Solver Logs

ASO writes a set of files for all Optimization problems within every Run executed with ASO at the following location:

<WLS_DOMAIN_HOME>/optresults/csv/<RUN_ID>/<opt problem number>_<fixture type>/

For example: <WLS_DOMAIN_HOME>/optresults/csv/1203/2105_shelf

In this directory are a set of directories and files containing all inputs and results for the last successful execution of that optimization problem. Each RUN_ID may contain many optimization problems.

<WLS_DOMAIN_HOME>/optresults/log/opt_problem_<RUN_ID>_<opt problem number>_*.*

For example: <WLS_DOMAIN_HOME>/optresults/log/opt_problem_1204_2113_287_.log

In this directory are a set of log files for each RUN_ID execution. Note a run can be executed many times so there may be multiple files.

The Gurobi solver also writes the following log:

<WLS_DOMAIN_HOME>/Gurobi.log - this file is useful for start & stop time and a summary of solver execution times.

# D

# Sample Category Management to ASO Import Files

This appendix contains some sample files in order to provide examples for reference.

### asrt_plan.txt

```
10000_2_q1_2014_1|10000|"Coffee - Northwest - 1st Qrtr,
FY2014"|Northwest|1|1|dest|"Assortment:Maintain,Space:Maintain"|
```

### cluster.txt

```
10000_2_q1_2014_1|102|Premium - Small - E (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|112|Value - Large - A (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|12|Mainstream - Large - A (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|122|Value - Medium - B (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|132|Value - Small - C (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|152|Value - Small - E (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|22|Mainstream - Medium - B (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|32|Mainstream - Small - C (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|42|Mainstream - Small - D (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|52|Mainstream - Small - E (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|62|Premium - Large - A (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|72|Premium - Medium - B (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|82|Premium - Small - C (Northwest)|2014-02-08|2015-01-31
10000_2_q1_2014_1|92|Premium - Small - D (Northwest)|2014-02-08|2015-01-31
```

### store.txt

```
10000_2_q1_2014_1|56|12||
10000_2_q1_2014_1|57|12||
10000_2_q1_2014_1|58|12||
10000_2_q1_2014_1|59|22||
10000_2_q1_2014_1|60|22||
10000_2_q1_2014_1|61|22||
10000_2_q1_2014_1|62|22||
10000_2_q1_2014_1|63|22||
10000_2_q1_2014_1|64|22||
10000_2_q1_2014_1|65|32||
10000_2_q1_2014_1|66|32||
10000_2_q1_2014_1|67|42||
10000_2_q1_2014_1|68|52||
10000_2_q1_2014_1|69|62||
10000_2_q1_2014_1|70|72||
10000_2_q1_2014_1|71|72||
10000_2_q1_2014_1|72|82||
10000_2_q1_2014_1|73|82||
10000_2_q1_2014_1|74|82||
10000_2_q1_2014_1|75|92||
10000_2_q1_2014_1|76|102||
10000_2_q1_2014_1|77|112||
10000_2_q1_2014_1|78|112||
10000_2_q1_2014_1|79|112||
10000_2_q1_2014_1|80|112||
10000_2_q1_2014_1|81|122||
10000_2_q1_2014_1|82|122||
10000_2_q1_2014_1|83|122||
10000_2_q1_2014_1|84|122||
10000_2_q1_2014_1|85|122||
10000_2_q1_2014_1|86|132||
10000_2_q1_2014_1|87|132||
10000_2_q1_2014_1|88|152||
10000_2_q1_2014_1|89|152||
10000_2_q1_2014_1|90|152||
10000_2_q1_2014_1|91|152||
10000_2_q1_2014_1|92|152||
```

### asrt_prod.txt

```
10000_2_q1_2014_1|102|1234582|0.50816055284631|-1
10000_2_q1_2014_1|102|1234600|0|-1
10000_2_q1_2014_1|102|1234615|0|-1
10000_2_q1_2014_1|102|1234747|0.72349577438432|-1
10000_2_q1_2014_1|102|1234753|1.0852560737945|-1
10000_2_q1_2014_1|102|1234759|0.83924281019|-1
10000_2_q1_2014_1|102|1234762|1.1286647031579|-1
10000_2_q1_2014_1|102|1234765|0.65114495572408|-1
10000_2_q1_2014_1|102|1234768|0.62877450858516|-1
10000_2_q1_2014_1|102|1234780|0|-1
10000_2_q1_2014_1|102|1234786|0|-1
10000_2_q1_2014_1|102|1234789|0|-1
10000_2_q1_2014_1|102|1234816|0.97811389479089|-1
10000_2_q1_2014_1|102|1234822|1.2226520289682|-1
10000_2_q1_2014_1|102|1234825|1.2575794928109|-1
10000_2_q1_2014_1|102|1234828|0|-1
10000_2_q1_2014_1|102|1234831|0|-1
10000_2_q1_2014_1|102|1234843|0.23225742301771|-1
10000_2_q1_2014_1|102|1234921|0|-1
10000_2_q1_2014_1|102|1234942|1.7277534856798|-1
10000_2_q1_2014_1|102|1234951|0.78406230072237|-1
10000_2_q1_2014_1|102|1234957|0.39361544408567|-1
10000_2_q1_2014_1|102|1235002|0.47538075503144|-1
10000_2_q1_2014_1|102|1235014|1.3953377336676|-1
10000_2_q1_2014_1|102|1235098|1.3575189435645|-1
10000_2_q1_2014_1|102|1235101|0|-1
10000_2_q1_2014_1|102|1235230|1.5341766210173|-1
10000_2_q1_2014_1|102|1235407|0.59201578717623|-1
10000_2_q1_2014_1|102|1235548|1.364893053759|-1
10000_2_q1_2014_1|102|1235563|1.2285012510976|-1
10000_2_q1_2014_1|102|1235572|1.4056739032405|-1
10000_2_q1_2014_1|102|1235575|1.2494945955836|-1
10000_2_q1_2014_1|102|1235581|0.7624202745485|-1
10000_2_q1_2014_1|102|1235596|1.17156534246|-1
10000_2_q1_2014_1|102|1235611|1.2917185341199|-1
10000_2_q1_2014_1|102|1235626|0.71583928232444|-1
10000_2_q1_2014_1|102|1235629|1.27799621557|-1
10000_2_q1_2014_1|102|1235632|1.1113098368409|-1
10000_2_q1_2014_1|102|1235674|1.2944721641804|-1
```

```
10000_2_q1_2014_1|102|1235701|1.3973811938995|-1
10000_2_q1_2014_1|102|1235716|1.4742074884796|-1
10000_2_q1_2014_1|102|1235719|1.4250759125732|-1
10000_2_q1_2014_1|102|1235728|0|-1
10000_2_q1_2014_1|102|1235737|0.89696401575458|-1
10000_2_q1_2014_1|102|1235752|1.1415226014793|-1
10000_2_q1_2014_1|102|1235785|1.1390854210803|-1
10000_2_q1_2014_1|102|1235794|1.2502182832382|-1
10000_2_q1_2014_1|102|1235806|1.1380997533479|-1
10000_2_q1_2014_1|102|1235842|1.4948738915156|-1
10000_2_q1_2014_1|102|1235845|1.2673979997101|-1
10000_2_q1_2014_1|102|1235854|1.5598690232596|-1
10000_2_q1_2014_1|102|1235884|0.48859925765996|-1
10000_2_q1_2014_1|102|1235908|0.54256433620554|-1
10000_2_q1_2014_1|102|1236016|0.62357105132935|-1
10000_2_q1_2014_1|102|1236040|0|-1
10000_2_q1_2014_1|102|1236166|0.54562496446729|-1
10000_2_q1_2014_1|102|1236190|0|-1
10000_2_q1_2014_1|102|1236193|1.0695049592627|-1
10000_2_q1_2014_1|102|1236199|1.2279523920662|-1
10000_2_q1_2014_1|102|1236205|1.0586334706756|-1
10000_2_q1_2014_1|102|1236214|0.70749671025385|-1
10000_2_q1_2014_1|102|1236229|0.87078098914116|-1
10000_2_q1_2014_1|102|1236235|0|-1
10000_2_q1_2014_1|102|1236238|0|-1
10000_2_q1_2014_1|102|1236367|0.081562743735713|-1
10000_2_q1_2014_1|102|1236487|0|-1
10000_2_q1_2014_1|102|1236496|0.84216650548228|-1
10000_2_q1_2014_1|102|1236511|1.6255380786619|-1
10000_2_q1_2014_1|102|1236544|0.73688932484641|-1
10000_2_q1_2014_1|102|1236640|0|-1
10000_2_q1_2014_1|102|1236664|0.26334390901703|-1
10000_2_q1_2014_1|102|1236808|0.72594453651108|-1
10000_2_q1_2014_1|102|1236823|0|-1
10000_2_q1_2014_1|102|1236832|1.3916467850777|-1
10000_2_q1_2014_1|102|1236841|1.1575944802367|-1
10000_2_q1_2014_1|102|1236847|0.940554436474|-1
10000_2_q1_2014_1|102|1236856|1.0563076783887|-1
10000_2_q1_2014_1|102|1236874|0|-1
10000_2_q1_2014_1|102|1236880|1.2863042699599|-1
10000_2_q1_2014_1|102|1234769|0.62877450858516|-1
10000_2_q1_2014_1|102|1234770|1.0200176405498|-1
10000_2_q1_2014_1|102|1234771|0.81041015663561|-1
10000_2_q1_2014_1|102|1234772|0.90824116564627|-1
10000_2_q1_2014_1|102|1234772|0.90824116564627|-1
10000_2_q1_2014_1|102|1234773|0.62877450858516|-1
10000_2_q1_2014_1|102|1234774|1.0937089066335|-1
10000_2_q1_2014_1|102|1234775|1.0937089066335|-1
10000_2_q1_2014_1|102|1234776|0|-1
10000_2_q1_2014_1|102|1234777|0|-1
10000_2_q1_2014_1|102|1234778|0|-1
10000_2_q1_2014_1|102|1234779|0|-1
10000_2_q1_2014_1|102|1234781|0|-1
10000_2_q1_2014_1|112|1234582|1.4895596077805|-1
10000_2_q1_2014_1|112|1234600|0|-1
10000_2_q1_2014_1|112|1234615|0|-1
```

# E

# Retail Analytics Interfaces Files

Note that ORASE consumes this data directly from RADM and is not needed if RADM is properly configured and populated. It is included purely for reference.

## Interfaces

### Product Hierarchy

The hierarchy that groups products into related groups.

*Table E–1 Product Hierarchy File*

| Field | Type | Description |
|---|---|---|
| Merchandise Hierarchy External ID | Text | Customer-visible Merchandise ID, used to denote any non-leaf node of the product hierarchy. (Source: lvl#anc_prodcat_id or top_lvl_prodcat_id from w_prod_cat_dh) |
| Merchandise Hierarchy PK | Text | Primary key used to identify a level of the merchandise hierarchy. (Source: ROW_WID from w_prod_cat_dh) |
| Merchandise Hierarchy External Key | Text | Externally suitable integration ID for the merchandise hierarchy. (Source integration_id from w_prod_cat_dh) |
| Effective From Date | Text | Starting date the source record is effective from. (Source: effective_from_dt in w_prod_cat_dh) |
| Effective To Date | Text | Ending date the source record is effective from. (Source: effective_to_dt in w_prod_cat_dh) |
| Current Flag | Text | Y/N flag to indicate if the record is considered a currently active hierarchy record. (Source: current_flg in w_prod_cat_dh) |
| Delete Flag | Text | Y/N flag to indicate if the record is considered deleted. (Source: delete_flg in w_prod_cat_dh) |
| Merchandise Hierarchy Name | Text | Name of the merchandise hierarchy. (Source: domain_member_name from w_domain_member_lkp_tl) |
| Merchandise Hierarchy Description | Text | Description of the merchandise hierarchy. (Source: domain_member_descr from w_domain_member_lkp_tl) |

## Product

Product is the terminal node of the product/merchandise hierarchy.

***Table E–2 Product File***

| Field | Type | Description |
|---|---|---|
| Merchandise External ID | Text | External Merchandise ID, used to denote the leaf node of the product hierarchy. (Source:prod_num from w_product_d) |
| Merchandise PK | Text | Primary key used to identify a SKU of the merchandise hierarchy. (Source: SCD1_WID from w_product_d) |
| Merchandise External Key | Text | Externally suitable integration ID for the merchandise hierarchy. (Source integration_id from w_product_d) |
| Parent Product Hierarchy ID | Text | FK to the parent hierarchy this product belongs to. (Source prod_cat5_wid from w_product_d) |
| Effective From Date | Date | Starting date the source record was effective from. (Source: effective_from_dt in w_product_d) |
| Effective To Date | Date | Ending date the source record was effective from. (Source: effective_to_dt in w_product_d) |
| Current Flag | Text | Y/N flag to indicate if the record is considered a currently active hierarchy record. (Source: current_flg in w_product_d) |
| Delete Flag | Text | Y/N flag to indicate if the record is considered deleted. (Source: delete_flg in w_product_d) |
| Merchandise Name | Text | Name of the SKU. (Source: product_name from w_product_d_tl) |
| Merchandise Description | Text | Description of the SKU. (Source: product_descr from w_product_d_tl) |

## Product Attributes

Generic attributes for products.

***Table E–3 Product Attributes File***

| Field | Type | Description |
|---|---|---|
| Merchandise PK | Text | Primary key used to identify a SKU of the merchandise hierarchy. |
| Attribute Number Value | Number | Numeric value for a generic product attribute. (Source: product_attr#_num_value from w_ product_attr_d) |
| Attribute Text Value | Text | Text value for a generic product attribute. (Source: product_attr#_name from w_product_ attr_d) |

## User Defined Product Attributes

User defined attributes for products.

*Table E–4 User Defined Product Attributes File*

| Field | Type | Description |
|---|---|---|
| Merchandise PK | Text | Primary key used to identify a SKU of the merchandise hierarchy.<br>(Source: prod_scd1_wid from w_rtl_item_grp1_d) |
| Attribute ID | Text | External ID for the UDA. (Source: flex_attrib_1_char from w_rtl_item_grp1_d) |
| Attribute Value ID | Text | External ID for the UDA value. (Source: flex_attrib_4_char from w_rtl_item_grp1_d) |
| Attribute Value Descr | Text | Descriptive value of the attribute value. (Source: Domain_member_name from w_rtl_domain_member_lkp_tl where domain_code = ITEM_UDA) |

## Organization Hierarchy

A hierarchy under which the organization can be organized.

*Table E–5 Organization Hierarchy File*

| Field | Type | Description |
|---|---|---|
| Organization Hierarchy External ID | Text | Customer visible ID used to identify an organization hierarchy level member. (Source: org_top_num/org_hier#_num columns from w_int_org_dh) |
| Organization Hierarchy PK | Text | Primary key used to identify a level of the organization hierarchy. (Source: scd1_wid from w_int_org_dh) |
| Organization Hierarchy External Key | Text | Externally suitable integration ID for the location hierarchy. (Source integration_id from w_int_org_dh) |
| Effective From Date | Date | Starting date the source record was effective from. (Source: effective_from_dt in w_int_org_dh) |
| Effective To Date | Date | Ending date the source record was effective from. (Source: effective_to_dt in w_int_org_dh) |
| Current Flag | Text | Y/N flag to indicate if the record is considered a currently active hierarchy record. (Source: current_flg in w_int_org_dh) |
| Delete Flag | Text | Y/N flag to indicate if the record is considered deleted. (Source: delete_flg in w_int_org_dh) |
| Organization Hierarchy Name | Text | Name of the organization hierarchy. (Source: domain_member_name from w_domain_memeber_lkp_tl) |
| Organization Hierarchy Description | Text | Description of the organization hierarchy. (Source: domain_member_descr from w_domain_member_lkp_tl) |

## Store Locations

Store locations is the terminal node of the organization hierarchy

*Table E–6 Store Locations File*

| Field | Type | Description |
|---|---|---|
| Location External ID | Text | Customer visible ID used to identify an organization location. (Source: Appropriate org_hier#_num column from w_int_org_dh) |
| Location PK | Text | Primary key used to identify a store of the location hierarchy. (Source: SCD1_WID from w_int_org_dh) |
| Location External Key | Text | Externally suitable integration ID for the location hierarchy. (Source integration_id from w_int_org_dh) |
| Parent Organization Hierarchy ID | Text | FK to the organization hierarchy this location belongs to. (Source: appropriate org_hier#_num column from w_int_org_dh) |
| Effective From Date | Date | Starting date the source record was effective from. (Source: effective_from_dt in w_int_org_dh) |
| Effective To Date | Date | Ending date the source record was effective from. (Source: effective_to_dt in w_int_org_dh) |
| Current Flag | Text | Y/N flag to indicate if the record is considered a currently active hierarchy record. (Source: current_flg in w_int_org_dh) |
| Delete Flag | Text | Y/N flag to indicate if the record is considered deleted. (Source: delete_flg in w_int_org_dh) |
| Location Name | Text | Name of the Location. (Source: org_name from w_int_org_d_tl) |
| Location Description | Text | Description of the Location. (Source: org_descr from w_int_org_d_tl) |

## Location Attributes

Generic attributes for locations.

*Table E–7 Location Attributes File*

| Field | Type | Description |
|---|---|---|
| Location PK | Text | Primary key used to identify a store of the location hierarchy. |
| Attribute Number Value | Number | Numeric value for a generic location attribute. (Source: org_attr#_num_value from w_int_org_attr_d) |
| Attribute Text Value | Text | Text value for a generic location attribute. (Source: org_attr#_name from w_int_org_attr_d) |

## Customer Segments

The list of customer segments that enable aggregate processing of customer data.

*Table E–8 Customer Segments File*

| Field | Type | Description |
|---|---|---|
| Customer Segment External ID | Text | Customer visible ID used to identify an customer segment. (Source: custseg_id from w_rtl_custseg_d) |
| Customer Segment PK | Text | Primary key used to identify a customer segment. (Source: row_wid from w_rtl_custseg_d) |
| Customer Segment External Key | Text | Externally suitable integration ID for the customer segment. (Source integration_id from w_rtl_custseg_d) |
| Effective From Date | Date | Starting date the source record was effective from. (Source: effective_from_dt in w_rtl_custseg_d) |
| Effective To Date | Date | Ending date the source record was effective from. (Source: effective_to_dt in w_rtl_custseg_d) |
| Current Flag | Text | Y/N flag to indicate if the record is considered a currently active hierarchy record. (Source: current_flg in w_rtl_custseg_d) |
| Delete Flag | Text | Y/N flag to indicate if the record is considered deleted. (Source: delete_flg in w_rtl_custseg_d) |
| Customer Segment Name | Text | Name of the customer segment. (Source: custseg_name from w_rtl_custseg_d) |

## Customer Segment Customer Members

Associates customer IDs to customer segment IDs, so that a customer's identity can be classified under a customer segment.

*Table E–9 Customer Segment Customer Members File*

| Field | Type | Description |
|---|---|---|
| Customer Segment ID | Text | ID of the customer segment (Source: custseg_hier_id from w_rtl_cust_custseg_d) |
| Customer ID | Text | ID of the customer (Source: customer_id from w_rtl_cust_custseg_d) |

## Customer Segment

This interface provides the hierarchy members for the Consumer Segment dimension.

*Table E–10 Customer Segment File*

| Field | Type | Description |
|---|---|---|
| Consumer Segment External ID | Text | Customer visible ID used to identify a Consumer Segment. |
| Consumer Segment PK | Text | Primary key used to identify a Consumer Segment. |
| Consumer Segment External Key | Text | Externally suitable integration ID for the Consumer Segment. |
| Effective From Date | Date | Starting date the source record was effective from. |
| Effective To Date | Date | Ending date the source record was effective from. |

***Table E–10 (Cont.) Customer Segment File***

| Field | Type | Description |
|---|---|---|
| Current Flag | Text | Y/N flag to indicate if the record is considered a currently active record. |
| Delete Flag | Text | Y/N flag to indicate if the record is considered deleted. |
| Consumer Segment Name | Text | Name of the Consumer Segment. |

## Consumer Segment/Location/Product Mapping

An allocation of store locations to any number of product hierarchy members per consumer segment.

***Table E–11 Consumer Segment/Location/Product Mapping File***

| Field | Type | Description |
|---|---|---|
| Consumer Segment PK | Text | Primary key of a consumer segment. |
| Location PK | Text | Primary key used to identify a store location. |
| Product Hierarchy PK | Text | Primary key used to identify a member of the product hierarchy. |
| Allocation Percent | Number | Percent of this location and product intersection that is allocated to this consumer segment. |

## Fiscal Year

Fiscal calendar year level data.

***Table E–12 Fiscal Year File***

| Field | Type | Description |
|---|---|---|
| Fiscal Year PK | Text | Primary Key suitable for identifying the fiscal year. (Source: row_wid from w_mcal_year_d) |
| Fiscal Year | Number | Externally known identifier for the fiscal year. (Source: mcal_year from w_mcal_year_d) |
| Fiscal Year Name | Text | The name associated with the fiscal year. (Source: mcal_per_name_year from w_mcal_year_d) |
| Start Date | Date | The date the fiscal year starts. (Source: mcal_year_start_dt from w_mcal_year_d) |
| End Date | Date | The date the fiscal year ends. (Source: mcal_year_end_dt from w_mcal_year_d) |

## Fiscal Quarter

Fiscal calendar's quarter level data.

*Table E–13 Fiscal Quarter File*

| Field | Type | Description |
|---|---|---|
| Fiscal Quarter PK | Text | Primary Key suitable for identifying the fiscal quarter. (Source: row_wid from w_mcal_qtr_d) |
| Fiscal Quarter | Number | Externally known identifier for the fiscal quarter. (Source: mcal_qtr from w_mcal_qtr_d) |
| Fiscal Quarter Name | Text | The name associated with the fiscal quarter. (Source: mcal_per_name_qtr from w_mcal_qtr_d) |
| Start Date | Date | The date the fiscal quarter starts. (Source: mcal_qtr_start_dt from w_mcal_qtr_d) |
| End Date | Date | The date the fiscal quarter ends. (Source: mcal_qtr_end_dt from w_mcal_qtr_d) |

## Fiscal Period

Fiscal calendar's period level data.

*Table E–14 Fiscal Period File*

| Field | Type | Description |
|---|---|---|
| Fiscal Period PK | Text | Primary Key suitable for identifying the fiscal period. (Source: row_wid from w_mcal_period_d) |
| Fiscal Period | Number | Externally known identifier for the fiscal period. (Source: mcal_period from w_mcal_period_d) |
| Fiscal Period Name | Text | The name associated with the fiscal period. (Source: mcal_per_name_period from w_mcal_period_d) |
| Start Date | Date | The date the fiscal period starts. (Source: mcal_period_start_dt from w_mcal_period_d |
| End Date | Date | The date the fiscal period ends. (Source: mcal_period_end_dt from w_mcal_period_d) |

## Fiscal Day

Fiscal Calendar day information.

*Table E–15 Fiscal Day File*

| Field | Type | Description |
|---|---|---|
| Fiscal Day PK | Text | Primary Key suitable for identifying the fiscal day. (Source: row_wid from w_mcal_day_d) |
| Fiscal Day | Number | Externally known identifier for the fiscal day. (Source: mcal_day_of_year from w_mcal_day_d) |
| Date | Date | The date of the fiscal day. (Source: mcal_day_dt from w_mcal_day_d) |

## Fiscal Week

Fiscal calendar week level data.

*Table E–16 Fiscal Week File*

| Field | Type | Description |
|---|---|---|
| Fiscal Week PK | Text | Primary Key suitable for identifying the fiscal week. (Source: row_wid from w_mcal_week_d) |
| Fiscal Week | Number | Externally known identifier for the fiscal week. (Source: mcal_week from w_mcal_week_d) |
| Fiscal Week Name | Text | The name associated with the fiscal week. (Source: mcal_per_name_week from w_mcal_week_d) |
| Start Date | Date | The date the fiscal week starts. (Source: mcal_week_start_dt from w_mcal_week_d) |
| End Date | Date | The date the fiscal week ends. (Source: mcal_week_end_dt from w_mcal_week_d) |

## Trade Area Hierarchy Levels

Provides the level descriptions for the trade area alternate location hierarchy.

*Table E–17 Trade Area Hierarchy Levels File*

| Field | Type | Description |
|---|---|---|
| Level Number | Number | A number that represents the level of the hierarchy. |
| Level Description | Text | Descriptive text for the hierarchy level. |
| Source Level Description | Text | The level description for the level of the location hierarchy this hierarchy converges with. |

## Trade Area Hierarchy

An alternate hierarchy for the organization hierarchy, which will group locations together by trading area.

*Table E–18 Trade Area Hierarchy File*

| Field | Type | Description |
|---|---|---|
| Trade Area Hierarchy External ID | Text | Customer visible ID used to identify a trade area hierarchy member. |
| Trade Area Hierarchy PK | Text | Primary key used to identify a level of the trade area hierarchy. |
| Trade Area Hierarchy External Key | Text | Externally suitable integration ID for the trade area hierarchy. |
| Effective From Date | Date | Starting date the source record was effective from. |
| Effective To Date | Date | Ending date the source record was effective from. |
| Current Flag | Text | Y/N flag to indicate if the record is considered a currently active hierarchy record. |

*Table E–18 (Cont.) Trade Area Hierarchy File*

| Field | Type | Description |
|---|---|---|
| Delete Flag | Text | Y/N flag to indicate if the record is considered deleted. |
| Trade Area Hierarchy Name | Text | Name of the trade area hierarchy. |
| Organization Hierarchy FK | Text | At the point this hierarchy converges with the organization hierarchy, a link to the organization hierarchy should be provided here. |

## Category Management Group Hierarchy

This is an alternate hierarchy for the product hierarchy, which can be used to group products together into groups that a customer would perceive as the same type of product.

*Table E–19 Category Management Group Hierarchy File*

| Field | Type | Description |
|---|---|---|
| Category Management Group Hierarchy External ID | Text | Customer visible Category Management Group ID, used to denote any non-leaf node of the Category Management Group hierarchy. (Source: lvl#anc_prodcat_id or top_lvl_prodcat_id from w_prod_cat_dh) |
| Category Management Group Hierarchy PK | Text | Primary key used to identify a level of the Category Management Group hierarchy. (Source: ROW_WID from w_prod_cat_dh) |
| Category Management Group Hierarchy External Key | Text | Externally suitable integration ID for the Category Management Group hierarchy. (Source integration_id from w_prod_cat_dh) |
| Effective From Date | Date | Starting date the source record was effective from. (Source: effective_from_dt in w_prod_cat_dh) |
| Effective To Date | Date | Ending date the source record was effective from. (Source: effective_to_dt in w_prod_cat_dh) |
| Current Flag | Text | Y/N flag to indicate if the record is considered a currently active hierarchy record. (Source: current_flg in w_prod_cat_dh) |
| Delete Flag | Text | Y/N flag to indicate if the record is considered deleted. (Source: delete_flg in w_prod_cat_dh) |
| Delete Flag<br>Category Management Group Hierarchy Name | Text | Name of the Category Management Group hierarchy. (Source: domain_member_name from w_domain_memeber_lkp_tl) |
| Category Management Group Hierarchy Description | Text | Description of the Category Management Group hierarchy. (Source: domain_member_descr from w_domain_member_lkp_tl) |
| Merchandise Hierarchy FK | Text | At the point this hierarchy converges with the merchandise hierarchy, a link to the merchandise hierarchy should be provided here. |

## Category Management Group Hierarchy Levels

This interface defines the levels that are part of the alternate category management group hierarchy.

*Table E–20 Category Management Group Hierarchy Levels File*

| Field | Type | Description |
|---|---|---|
| Level Number | Number | A number that represents the level of the hierarchy. |
| Level Description | Text | Descriptive text for the hierarchy level. |
| Source Level Description | Text | The level description that this alternate hierarchy converges with the primary product hierarchy. |

## Location/Product Price and Cost

Current (at a given date) price and cost information provided at the product store level.

*Table E–21 Location/Product Price and Cost File*

| Field | Type | Description |
|---|---|---|
| Product PK | Text | Primary key used to identify a product. |
| Location PK | Text | Primary key used to identify a store location. |
| Price | Number | Available price for the given date for the product/store combination. |
| Cost | Number | Available cost for the given date for the product/store combination. |

## Sales Transaction Data

Sales data at the product, location, day, and transaction level.

*Table E–22 Sales Transaction Data File*

| Field | Type | Description |
|---|---|---|
| Product PK | Text | Primary key used to identify a product.<br>(Source: prod_scd1_wid in w_rtl_sls_trx_it_lc_dy_f) |
| Location PK | Text | Primary key used to identify a store location.<br>(Source: org_scd1_wid in w_rtl_sls_trx_it_lc_dy_f) |
| Day PK | Number | Primary key used to identify the date of the transaction.<br>(Source: dt_wid in w_rtl_sls_trx_it_lc_dy_f) |
| Transaction ID | Text | Unique transaction identifier for a collection of sales for a given sales transaction.<br>(Source: sls_trx_id in w_rtl_sls_trx_it_lc_dy_f) |
| Sales Amount | | Amount of sales expressed in global currency<br>(Source: Sls_amt_lcl / loc_exchange_rate in w_sls_trx_it_lc_dy_f) |
| Sales Units | | Units sold.<br>(Source: sls_qty in w_rtl_sls_trx_it_lc_dy_f) |

*Table E–22 (Cont.) Sales Transaction Data File*

| Field | Type | Description |
|---|---|---|
| Sales Profit | | Profit amount expressed in global currency<br>(Source: sls_profit_amt_lcl / loc_exchange_rate in w_rtl_sls_trx_it_lc_dy_f) |
| Customer ID | Text | PK for the customer who completed the transaction.<br>(Source: customer_wid in w_rtl_sls_trx_it_lc_dy_f) |
| Promotion ID | Text | Primary Key for the promotion component for this sales record.<br>(Source: promo_comp_wid in w_rtl_slspr_trx_it_lc_dy_f) |

## Item Ranging

Source: W_RTL_IT_LC_D table.

*Table E–23 Item Ranging File*

| Field | Type | Description |
|---|---|---|
| Product ID | VARCHAR2(30) | This is a foreign key to the W_INT_ORG_D table using the SCD Type 1 WID.<br>(Source: PRODUCT_SCD1_WID) |
| Location ID | VARCHAR2(30) | This is a foreign key to the W_INT_ORG_D table using the SCD Type 1 WID.<br>(Source: ORG_SCD1_WID |
| Ranged indicator | VARCHAR2(30) | This is the indicator from the source table that states that even though the item is not present in a particular location it exists in inventory.<br>This assists in acceptance of item not ranged in a location to be accepted in case of customer or cross channel returns.<br>(Source: RANGED) |
| Status of the item | VARCHAR2(1) | Current status of item at the store. Valid values are: A = Active (item is valid and can be ordered) and sold, I = Inactive (item is valid but cannot be ordered) or sold, C = Discontinued (item is valid and sellable but no longer order-able, and D = Delete (item is invalid and cannot be ordered or sold).<br>(Source: ITEM_STATUS) |
| Location type | VARCHAR2(1) | Type of location in the location field. Valid values are S (store), W (warehouse), and E (external finisher).<br>(Source: LOC_TYPE) |
| Start effective date | Date | (Source: EFFECTIVE_FROM_DT) |
| End effective date | Date | (Source: EFFECTIVE_TO_DT) |

# F

# Sample Application Context File (DT)

This appendix contains portions of a sample DT middle-tier configuration file in order to provide examples for reference.

### dttojp-application-context.xml

```
<!-- allowed parameters along with types -->
<bean id="param" class="oracle.retail.rse.common.jobprocessor.Param"
    scope="singleton">
    <constructor-arg>
        <map>
            <entry key="batchId"><value>Long</value></entry>
            <entry key="stageId"><value>Integer</value></entry>
            <entry key="userId"><value>String</value></entry>
            <entry key="versionId"><value>Long</value></entry>
            <entry key="cmGrpId"><value>Long</value></entry>
            <entry key="idForSubstep"><value>Long</value></entry>
            <entry key="srvcType"><value>String</value></entry>
            <entry key="reqQueueId"><value>Long</value></entry>
            <entry key="appName"><value>String</value></entry>
            <entry key="procName"><value>String</value></entry>
            <entry key="timeout"><value>Integer</value></entry>
            <entry key="waitTime"><value>Integer</value></entry>
            <entry key="scnConfigHdrParamId"><value>Long</value></entry>
            <entry key="scnConfigHdrParamIdStr"><value>String</value></entry>
        </map>
    </constructor-arg>
</bean>

<!-- refered from RSE_PROC_TASK_TMPL.CTL -->
<bean id="dt" class="oracle.retail.rse.cdm.dt.fe.jobprocessoradapter.DtService"
    scope="singleton">
    <constructor-arg> <!-- available parameters to set are batch/stage/userId -->
        <map> <!-- the first ? must be an OUT and return the starting stage id -->
            <entry key="0"><bean
class="oracle.retail.rse.common.jobprocessor.PlsqlConfig">
                <constructor-arg >
                <value><![CDATA[declare v_batch_id NUMBER := ?;
                    v_target_stage_id NUMBER := ?;
                    v_srvc_type VARCHAR2(30) := ?;
                    v_starting_stage_id NUMBER; begin
                v_starting_stage_id :=
…

<!-- refered from RSE_PROC_TASK_TMPL.CTL -->
<bean id="dtLocalSynchConcurrentServiceNoDbLog"
    class="oracle.retail.rse.common.jobprocessor.service.LocalServiceAdapter"
    scope="singleton" lazy-init="true" destroy-method="close">
        <constructor-arg name="logInDb" value="false"/>
        <constructor-arg name="useCurrentThread" value="true"/>
        <constructor-arg name="fixedThreadPoolSize" value="3"/>
</bean>

<!-- refered from RSE_PROC_TASK_TMPL.CTL - step processing. useCurrentThread true
means that the current thread is utilized -->
<bean id="dtLocalSynchConcurrentService"
class="oracle.retail.rse.common.jobprocessor.service.LocalServiceAdapter"
scope="singleton" lazy-init="true" destroy-method="close">
        <constructor-arg name="logInDb" value="true"/>
        <constructor-arg name="useCurrentThread" value="true"/>
        <constructor-arg name="fixedThreadPoolSize" value="3"/>
</bean>

</beans>
```

# Glossary of Acronyms

### AC

Advanced Clustering, a component of ORME, also known as CIS.

### ASO

Oracle Retail Assortment and Space Optimization.

### BI

Business Intelligence.

### CDT

Customer Decision Tree, a component of ORME.

### CM

Oracle Retail Category Management.

### DB

Database.

### DT

Demand Transference, a component of ORME.

### MBA

Market Basket Analysis.

### MDS

Oracle MetaData Services.

### ORASE

Oracle Retail Advanced Science Engine, also known as RSE. It contains CDT, DT, AC, MBA. and ASO.

### RA

Retail Analytics.

### RADM

Retail Analytics Data Model, also known as RA Schema.

### RDF

Oracle Retail Demand Forecasting.

### UI

User Interface.