Oracle® Retail Promotion Intelligence and Promotion Planning and Optimization

Operations Guide

Release 13.0.2

October 2008

ORACLE®

Oracle Retail Promotion Intelligence and Promotion Planning and Optimization Operations Guide, Release 13.0.2

Primary Author: Judith Meskill

Contributing Author:

Contributor:

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

(i) the software component known as **ACUMATE** developed and licensed by Lucent Technologies Inc. of Murray Hill, New Jersey, to Oracle and imbedded in the Oracle Retail Predictive Application Server - Enterprise Engine, Oracle Retail Category Management, Oracle Retail Item Planning, Oracle Retail Merchandise Financial Planning, Oracle Retail Advanced Inventory Planning and Oracle Retail Demand Forecasting applications.

(ii) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(iii) the **SeeBeyond** component developed and licensed by Sun MicroSystems, Inc. (Sun) of Santa Clara, California, to Oracle and imbedded in the Oracle Retail Integration Bus application.

(iv) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Store Inventory Management.

(v) the software component known as **Crystal Enterprise Professional and/or Crystal Reports Professional** licensed by Business Objects Software Limited ("Business Objects") and imbedded in Oracle Retail Store Inventory Management.

(vi) the software component known as **Access Via™** licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(vii) the software component known as **Adobe Flex™** licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

(viii) the software component known as **Style Report™** developed and licensed by InetSoft Technology Corp. of Piscataway, New Jersey, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

(ix) the software component known as **DataBeacon™** developed and licensed by Cognos Incorporated of Ottawa, Ontario, Canada, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

Contents

4 Standard Load

5 Historical Analysis

6 Predictive Modeling

7 Affinity Modeling

8 Export API

9 Tasks and Techniques

10 Technical Reference

A Sample Dataset

B Managing Your Applications

Preface

Oracle Retail Promotion Intelligence analyzes the results of past promotions and advertising and the affinity effects of products on one another to deliver insight into the performance of a promotional strategy.

Oracle Retail Promotion Planning and Optimization assists you in creating and improving your promotions. It allows you to leverage the information gained from Promotion Intelligence to make the best promotion decisions by using what-if analysis and predictive forecasting.

Promotion Planning and Optimization combines analysis, planning, and implementation components to give retailers the capability to achieve the highest return on their advertising, promotion, and inventory investments.

Audience

This document is intended for administrators of the Oracle Retail Promotion Intelligence and Promotion Planning and Optimization application.

Related Documents

For more information, see the following documents in the Oracle Retail Promote documentation set:

- *Oracle Retail Promotion Intelligence and Promotion Planning and Optimization Release Notes*
- *Oracle Retail Promotion Intelligence and Promotion Planning and Optimization Configuration Guide*
- *Oracle Retail Promotion Intelligence User Guide*
- *Oracle Retail Promotion Planning and Optimization User Guide*
- *Oracle Retail Promotion Intelligence and Promotion Planning and Optimization Installation Guide*

Customer Support

- https://metalink.oracle.com

When contacting Customer Support, please provide:

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

Review Patch Documentation

For a base release (".0" release, such as 13.0), Oracle Retail strongly recommends that you read all patch documentation before you begin installation procedures. Patch documentation can contain critical information related to the base release, based on new information and code changes that have been made since the base release.

Oracle Retail Documentation on the Oracle Technology Network

In addition to being packaged with each product release (on the base or patch level), all Oracle Retail documentation is available on the following Web site:

http://www.oracle.com/technology/documentation/oracle_retail.html

Documentation should be available on this Web site within a month after a product release. Note that documentation is always available with the packaged code on the release date.

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

The chapter contains the following:

About this Operations Guide

The *Promotion Intelligence and Promotion Planning and Optimization Operations Guide* provides details about the essential tasks involved in using the application: the staging and loading of data that is provided by the customer, historical analysis, market basket analysis, predictive modeling, and affinity modeling. For further information, see the *Promotion Intelligence and Promotion Planning and Optimization Configuration Guide*.

What's In This Book

The Promote Operations Guide addresses the following topics:

- Standard Interface
- Standard Load
- Historical Analysis
- Market Basket Analysis
- Predictive Modeling
- Affinity Modeling
- Export API
- Technical Reference
- Sample Dataset
- Managing the Application

2

Overview

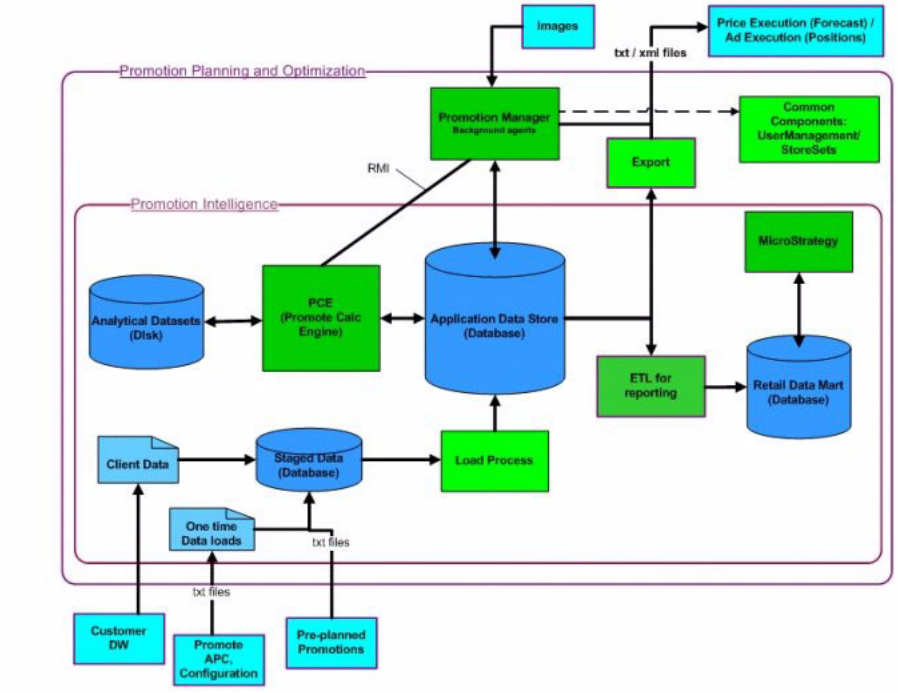
This chapter contains the following:

Introduction

This chapter provides an overview of the Promote architecture and analytic processes.

Architecture

The Promote application consists of the modules shown in the following diagram and discussed below:



The Promote modules are:

- Promotion Manager – the Promotion Planning and Optimization (PPO) application, which provides end user access to the Promote functionality, including the promotional calendar and what-if analysis.
- Promote Calc Engine (PCE), which is responsible for analytical calculations and modeling.
- A group of Analytical Datasets, including Baseline dataset, Ad Modeling dataset, and Product Affinities dataset. The PCE stores its analytical data in datasets.
- Application Database, including the Retail Data Mart (RDM)
- MicroStrategy, which provides access to reports.

Promote system inputs include:

- Transaction-level sales data
- Data feeds (described in Chapter 3, "Standard Interface")

Promote data is output to the following systems:

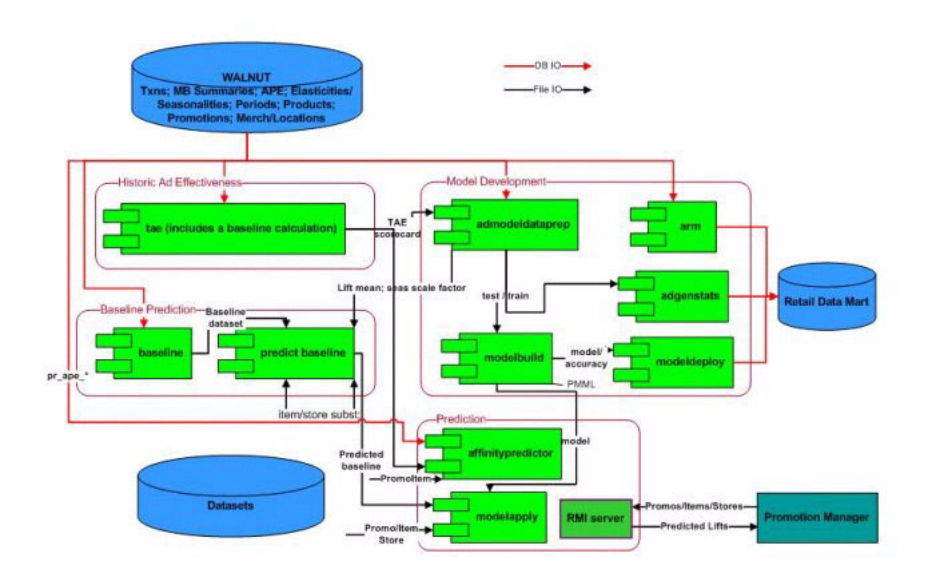
- Pricing Systems, which need promotion prices in order to send the correct data to stores.
- Merchandising and Replenishment Systems, which need information about the predicted unit lifts associated with a promotion to help order the correct amount of merchandise to support the promotion event.
- Ad Execution systems, which need the information to support the design and distribution of the advertising layouts that will be sent to a printer.

Promote Calc Engine

Promote's functionality is driven by analytics. This section provides details about the promote Calc Engine (PCE) and its analytical processes. For more information about the PCE, see Chapter 6, "Predictive Modeling".

Overview of the PCE Analytical Tools

The PCE consists of a group of tools that perform analytical operations, as shown in the following diagram:



The tools include:

- Baseline – helps the analyst construct a picture of the non-promoted performance of merchandise. The script baseline.sh is discussed in Chapter 5, "Historical Analysis".
- TAE – used to analyze and assess the performance and cross-product effect of an advertisement. The script tae.sh is discussed in Chapter 5, "Historical Analysis".
- Model Data Preparation – used to prepare data necessary for model construction. The model_prepare.sh script is discussed in Chapter 10, "Technical Reference".
- Model build – used to build a promotion performance model. The model_build.sh script is discussed in Chapter 6, "Predictive Modeling".
- ARM – performs association rule mining, which lets analysts discover relationships between products. The arm.sh script is discussed in Chapter 5, "Historical Analysis".
- Predict Baseline – used to build a dataset of an item's baseline sales for a future time period. The predict_baseline.sh script is discussed in Chapter 6, "Predictive Modeling".

The Promote application includes additional tools that support such analytical tasks as moving data between the database and datasets and managing datasets. These tools are discussed in Chapter 10, "Technical Reference".

Overview of Analytic Process

The analytical process consists of the following high-level steps:

1. **Estimate Price Elasticity and Seasonality Factors**
   Prior to using the baseline measurement tools, estimate the price elasticity and seasonality analytical factors.

2. **Baseline Measurement**
   Use baseline.sh to develop baseline measurements (sales absent promotions) for items.
3. **TAE Analysis**
   Use tae.sh to measure the effectiveness of all past promotions. Use tae_load.sh to load the TAE results (scorecards) into the RDM. Reports that can be used to analyze the information can be generated from the data in the RDM.
4. **ARM Analysis**
   Use arm.sh to create association metrics. Then use arm_stage.sh to load the association metrics into the database.
5. **Perform Market Basket Analysis**
   Use pl_create_summary_tables.sh, pl_create_summaries.sh, pl_create_promote_ir_views.sh, and pl_create_rdm_summary.sh to summarize the Market Basket data. Then, use pl_load_promote_rdm.sh to load Market Basket summary data into the RDM.
6. **Build Predictive Models**
   Predictive models are developed for different parts of the merchandise hierarchy. First, create the necessary data for each predictive model, using model_prepare.sh. For each model, use the model_build.sh tool iteratively to determine which attributes are significant.
7. **Estimate Affinity Parameters**
   Configure the Affinity Parameter Estimator (APE) tree and load it into the application. Then, use the APE to estimate the affinity parameters and load the data into the database.

Details about each of these steps are the subject of this guide.

3

Standard Interface

This chapter contains the following:

Introduction

An important part of getting Promote up and running in a production environment is the gathering and loading of enterprise data. Promote requires historical and weekly

data to be loaded into the Promote database. The data must be provided in a standard format, as specified in the standard interface specification. The data can then be loaded according to the standard load procedure.

This chapter contains the standard interface specifications for the data that is loaded into Promote.

Promote Standard Interface Descriptions

This section details the data interface to the Promote application. The interfaces are described in alphabetical order.

Promote requires that customer data be provided in flat files containing pipe-delimited data organized so that the data can be loaded into Promote database tables that follow the formats specified here.

The following special characters are not allowed: colon, semi-colon, comma, forward slash, backward slash, any type of quote, any type of apostrophe, <, or >.

Three interfaces (Merchandise Hierarchy Levels, Location Hierarchy Levels, and Cross Product Information) that are required by Promote are only loaded once. The information contained in these three files is collected during discussions with specific clients; however, the files themselves are not provided by clients but are created and loaded as part of the initial Promote configuration. More information on these three interfaces is provided in Chapter 4, “Standard Load.”

The standard interface includes the following:

Table 3–1 Interface Specifications

Interface Specification	Required/Optional	Needed for PI-only Implementation?
APE Price Elasticity	Optional	No
APE Promotion Elasticity	Optional	No
Calendar	Required	Yes
Cross Products Information – described in Chapter 4, “Standard Load.”	Required	Yes
Demand Parameters	Required	Yes
Future Price Cost	Optional	No
Images	Optional	No
Inventory	Required	Yes
Items	Required	No
Like Location	Optional	No
Like Merchandise	Optional	Yes
Location Hierarchy	Required	Yes
Location Hierarchy Attributes	Required	Yes
Location Hierarchy CDA	Optional	No
Location Hierarchy Levels – described in Chapter 4, “Standard Load.”	Required	Yes
Location Hierarchy Rename	Optional	Yes
Merchandise Hierarchy	Required	Yes

Table 3–1 (Cont.) Interface Specifications

Interface Specification	Required/Optional	Needed for PI-only Implementation?
Merchandise Hierarchy Attributes	Required	Yes
Merchandise Hierarchy CDA	Optional	No
Merchandise Hierarchy Levels – described in Chapter 4, "Standard Load."	Required	Yes
Merchandise Hierarchy Rename	Optional	Yes
Merchandise Thresholds	Required	No
Offers	Required	Yes
Period Attributes	Required	Yes
Promotion Allocation	Optional	Yes
Promotion Campaign	Optional	Yes
Promotion Forecaster	Required	No
Promotion Offer	Required	Yes
Promotion Offer Attributes	Required	Yes
Promotion Offer Criteria	Required	Yes
Promotion Offer Merchandise	Required	Yes
Promotion Offer Store	Required	Yes
Promotions	Required	Yes
Seasonal Trend	Required	Yes
Seasonalities	Required	Yes
SKU List	Required	No
SKU List Items	Required	No
Store Sets	Required	No
Store Set Prices	Required	No
Store Subsets	Required	No
Store Subset Assignments	Required	No
TAE Temp Metrics	Optional	Yes
Transaction Log	Required	Yes
UDE Type	Required	Yes
UDE Value	Required	Yes
Vehicle	Required	Yes
Vehicle Attributes	Required	Yes

APE Price Elasticity Standard Interface Description

The APE price elasticity interface describes the APE price elasticity data generated by the Affinity Parameter Estimator (APE) component of Promote.

Data Fields

Five fields describe each record:

- DRIVER_APE_MERCH_NODE_EXT_ID – the external ID for the Driver Merchandise node.
- TARGET_APE_MERCH_NODE_EXT_ID – the external ID for the Target Merchandise node.
- LOC_LEVEL_DESC – the external ID for the external location level.
- LOC_CLIENT_LOAD_ID – the external ID for the location.
- ELASTICITY – the APE-calculated elasticity value.

An Example

The following table shows sample APE Price Elasticity data.

Table 3–2 Sample APE Price Elasticity Data

Driver	Target	Location Level	Location ID	Elasticity
Toys: HIER3_KEY=1 80: HIER4_KEY=2 17: HIER5_KEY=3 17020:	Toys: HIER3_KEY=1 80: HIER4_KEY=2 17: HIER5_KEY=3 17023:	STORE	3451	0.4907

APE Promotion Elasticity Standard Interface Description

The APE price elasticity interface describes the APE promotion elasticity data generated by the Affinity Parameter Estimator (APE) component of Promote.

Data Fields

Six fields describe each record:

- DRIVER_APE_MERCH_NODE_EXT_ID – the external ID for the Driver Merchandise node.
- TARGET_APE_MERCH_NODE_EXT_ID – the external ID for the Target Merchandise node.
- LOC_LEVEL_DESC – the external ID for the external location level.
- LOC_CLIENT_LOAD_ID – the external ID for the location.
- PROMOTION_EXTERNAL_ATTR – a value generated by concatenating the source column name and its corresponding value.
- ELASTICITY – the APE-calculated elasticity value.

An Example

The following table shows sample APE Promotion Elasticity data.

Table 3–3 Sample APE Promotion Elasticity Data

Driver	Target	Location Level	Location ID	External Attribute	Elasticity
Toys: HIER3_KEY=1 80 HIER4_KEY=2 17 HIER5_KEY=3 17020	Toys: HIER3_KEY=1 80 HIER4_KEY=2 17 HIER5_KEY=3 17023	STORE	3451	VEHICLE:vehicle .circular	0.4907

Calendar Standard Interface Description

The calendar interface describes a retailer's fiscal calendar. Each record in the file corresponds to a single fiscal week.

Data Fields

Seven fields describe each calendar record, which represents a fiscal week:

- EOP_CALENDAR_DT – the last day of the fiscal week, which is usually Saturday.
- FISCAL_YR – the number of the fiscal year for the record.
- FISCAL_QTR – the number of the fiscal quarter for the record.
- FISCAL_MO – the number of the fiscal month for the record.
- FISCAL_WK – the number of the fiscal week for the record.
- CALENDAR_WK – an alternative number for the calendar week for the record.
- SEASON – the number identifying the season associated with the calendar week.

An Example

The following table shows sample data for five weeks of a fiscal calendar.

Table 3–4 Sample Calendar Data

EOP Calendar Date	Fiscal Year	Fiscal Quarter	Fiscal Month	Fiscal Week	Calendar Week	Season
2004-02-07	2004	1	1	1	1	1
2004-02-14	2004	1	1	2	2	1
2004-02-21	2004	1	1	3	3	1
2004-02-28	2004	1	1	4	4	1
2004-03-06	2004	1	2	5	1	1

Technical Notes

The following list provides details to consider regarding the calendar data.

- The calendar must include all weeks, beginning with the earliest historical sales record and extending at least two years into the future.
- Each year included in the data must contain 52 – 53 weeks.

- The calendar file can be sent weekly or loaded all at once during the initial configuration of Promote. If provided all at once, it should contain all the historic data and extend at least three years into the future.
- Retailers can use the SEASON field to designate different seasons within the fiscal year. For example, a retailer might divide the fiscal year into two seasons.

Demand Parameters Standard Interface

The demand parameters standard interface describes the mapping between the analytical parameter values generated by Analytical Services and a specific merchandise/location/attribute.

Data Fields

Nine fields describe each demand parameter record:

- MERCHANDISE_LEVEL – the external merchandise level.
- MERCHANDISE_KEY – the key from the merchandise hierarchy for the item.
- LOCATION_LEVEL – the external location level.
- LOCATION_KEY – the key from the location hierarchy for the item.
- ITEM_ATTRIBUTE – the item attribute for the parameter (set to% by default).
- PARAMETER_NAME – the name of the parameter. The names can be DEFAULT_GAMMA, DEFAULT_ALPHA, CRITICAL_INVENTORY, or ZERO_INVENTORY.
- PARAMETER_VALUE – the value assigned to the parameter.
- AS_PARAMETER_ID – a number that uniquely identifies the record across all output tables and can be used to trace issues. It is not an analytical value.
- AS_VERSION_NUMBER – the version number for the current run of the output, which is set by APC and can be used to track versions.

Future Price and Cost Standard Interface Description

The future price and cost interface describes future changes for price and cost. Data must be configured at the SKU level for merchandise and at the CHAIN level for location.

Data Fields

Seven fields describe each future price and cost record:

- MERCH_CLIENT_LOAD_ID – the customer's merchandise ID.
- MERCH_LEVEL_DESC – the merchandise level description.
- LOC_CLIENT_LOAD_ID – the customer's location ID.
- LOC_LEVEL_DESC – the location level description.
- EFFECTIVE_DT – the date of the change.
- PRICE – the changed price.
- COST – the changed cost.

An Example

The following is an example of the data for a future price and cost record:

Table 3–5 Future Price and Cost Example Data

Merch Client Load ID	Merch Level Desc	Loc Client Load ID	Loc Level Desc	Effective Dt	Price	Cost
T0000011506	SKU	0	CHAIN	2006-04-06	23.29	12.35

Images Standard Interface Description

The images interface describes the data feed that is used by clients to import their image library. Promote maintains a catalog of references to the images, not the images themselves.

Data Fields

Thirteen fields describe an images record:

- NAME – the display name for the image.
- EXTERNAL_NAME – the ID for the image that is meaningful to the client. It is unique across all images.
- DESCRIPTION – an optional description of the image.
- FILE_NAME – the filename for the image.
- KEYWORDS – keywords placeholder.
- FILE_SIZE – the size of the image file.
- WIDTH – the image width.
- HEIGHT – the image height.
- RESOLUTION – the on-screen resolution of the image.
- DEPTH – the depth of the image.
- FILE_TYPE_ENUM – the image file type. Must be JPEG (0).
- MERCH_CLIENT_LOAD_ID – the client-specific category ID.
- LEVEL_DESC – the client-specific merchandise hierarchy level description.

An Example

The following is an example of the data for an images record.

Table 3–6 Images Example Data

Name	External Name	Description	File Name	Keywords	File Size	Width	Height	Resolution	Depth	File Type Enum	Merch Client Load ID	Level Desc
CG Barbie Convertible	barbie caligirl convertible	Barbie car	barbie cgconvertible.jpg	barbie	1024	30	40			0	T00000 8493	SKU

Inventory Standard Interface Description

The inventory interface describes a client's historical inventory data. This data feed is used for loading the data used by the Affinity Parameter Estimator (APE) component. Promote requires the first five fields.

Data Fields

Twenty seven fields describe an inventory record:

- MERCHANDISE_KEY – the key from the merchandise hierarchy for the item. All items must be at the same level in the merchandise hierarchy, which for Promote is the Item level.
- LOCATION_KEY – the key from the location hierarchy for the item. All items must be at the same level in the location hierarchy, which for Promote is the Store level.
- FISCAL_YR – the fiscal year of the sales record.
- FISCAL_WK – the fiscal week of the sales record.
- END_OH_QTY – the number of units of on-hand inventory at the end of the period.
- END_OO_QTY – the number of inventory units in transit to the location at the end of the period.
- UNIT_RTL – the item's ticketed price at the end of the period.
- UNIT_CST – the item's unit cost at the end of the period.
- INIT_RTL – the item's ticketed price at the start of the season.
- RECEIPT_QTY – the total store receipts (in units) from the distribution centers and from transfers.
- GRSS_SLS_QTY – the gross number of new units sold for the item at the location. This excludes returns.
- GRSS_SLS_AMT – the gross dollar amount of new sales for the item at the location during the period. This excludes returns.
- NET_SLS_QTY – the net number of units sold of the item at the location. This includes returns.
- NET_SLS_AMT – the net dollar amount of sales for the item at the location during the period. This includes returns.
- TOT_DSC_AMT – the total discount amount.
- PROMO_MKDN_DSC_AMT – the total promotional markdown discount amount.
- SELLIT_MKDN_DSC_AMT – the total sell-it discount amount.
- CLR_DSC_AMT – the total clearance discount amount.
- FREIGHT – the freight cost.
- GRSS_PROFIT_AMT – the total gross margin (profit).
- DUMMY – a dummy field.
- POS_SLS_QTY – the number of new units sold of the item at the location during the period.
- POS_SLS_AMT – the dollar amount of the new sales for the item at the location during the period.

- MD_SALES_QTY – the units sold while on markdown.
- MD_SALES_AMT – the sales dollars of the units sold while on markdown.
- POS_MD_AMT – the total difference in weekly sales dollars between the promotional sales price and the inventory price.
- PERM_MD_AMT – includes distribution center, on hand, in transit, and store on hand.

An Example

The following is an example of the data for an inventory record. Only the first five fields, which are required, are shown.

Table 3–7 Inventory Example Data

Merchandise Key	Location Key	Fiscal Yr	Fiscal Wk	End OH Qty
T0000084953	5773	2004	9	2568

Items Standard Interface

The items interface describes valid combinations of merchandise and location that specify an item. All items are defined at a single level of the merchandise hierarchy (typically the lowest level) and a single level in the location hierarchy. For the merchandise and location hierarchy examples provided, items might be defined as combinations of Style in the merchandise hierarchy and Region in the location hierarchy. (For information about the configuration of the hierarchy levels that define items, see Chapter 4, "Standard Load")

Data Fields

Nine fields describe an item:

- MERCHANDISE_KEY – the key from the merchandise hierarchy for the item. (All items must be at the same level in the merchandise hierarchy.)
- LOCATION_KEY – the key from the location hierarchy for the item. (All items must be at the same level in the location hierarchy.)
- FIRST_RECEIPT_DATE – the date of the first receipt of this merchandise at this location. This date, if available, defines the beginning of life for an item. Several
- LAST_RECEIPT_DATE – the date of the most recent receipt of this item at the item's location.
- VENDOR – the supplier for the item.
- VENDOR_DESC – a description of the supplier.
- UNIT_COST – the average unit cost of the item.
- SEASON_CODE – a retailer-specific code that can be used to help determine an item's seasonality. For example, a retailer may have four season codes (Spring, Summer, Fall, and Winter), and the seasonality assignment may be based on merchandise class and season code. Alternatively, some retailers may supply a Floor Set or Store Layout code in this field if such data exists. This may be more relevant for determining seasonality.
- FULL_PRICE – the original retail price of the merchandise.

An Example

The following table shows sample items, based on the sample data provided in the Merchandise Hierarchy and Location Hierarchy sections.

Table 3–8 Items Sample Data

Merchandise Key	Location Key	First Receipt Date	Last Receipt Date	Vendor	Vendor Desc.	Unit Cost	Season Code	Full Price
101234509	FL1	2004-11-07	2004-11-21			9.53	Fall3	14.99
101234509	FL2	2004-10-31	2004-11-07			9.98	Fall2	15.99
101234512	O1	2005-01-24	2005-01-24			17.40	Spring1	24.99
101234512	O2	2005-01-31	2005-01-31			17.40	Spring1	24.99

The items in the example are defined at the Color-Region level. For example, the first item is color 101234509 and region FL1. It is possible for items with the same product key to have different values for other fields. The same piece of merchandise may have different cost, vendor, receipt date, or season code values for different locations. In addition, a single piece of merchandise may not be defined as a valid item for all locations.

Like Location Standard Interface Description

The like location interface describes the association between a store and a similar store. The data feed can be used to add or remove associations.

Data Fields

Four fields describe a like location record:

- LOC_CLIENT_LOAD_ID – the customer's location ID for the location without promotion history.
- LOC_LEVEL_DESC – the location level description.
- LIKE_LOC_CLIENT_LOAD_ID – the customer's like location ID for the location with promotion history information available. These attributes are used for the substitution.
- LIKE_LOC_LEVEL_DESC – the like location level description.

Like Merchandise Standard Interface Description

The like merchandise interface describes the association between an item and a similar item. The data feed can be used to add or remove associations. Note that since the data feed can remove most entries in the target table, it is expected that a user will either use the data feed exclusively or the Promote UI exclusively. (In either case, the data feed can be used to initially set up the system.) Data can be loaded incrementally; a full refresh is not necessary. The INACTIVE flag for rows to be removed must be set to 1 (Inactive).

Data Feeds

Five fields describe a like merchandise record:

- MERCH_CLIENT_LOAD_ID – the customer's merchandise ID for merchandise without promotion history.
- MERCH_LEVEL_DESC – the merchandise level description.

- LIKE_MERCH_CLIENT_LOAD_ID – the customer's like merchandise ID for merchandise with promotion history available. These attributes are used for the substitution.
- LIKE_MERCH_LEVEL_DESC – the like merchandise level description.
- INACTIVE - status of the data that is used to identify whether or not the data is to be added or removed. Values are 0 = Active and 1 = Inactive. The flag for items to be removed must be set to 1.

Location Hierarchy Standard Interface Description

The location hierarchy interface describes how a retailer categorizes locations. The location hierarchy begins with the highest level, such as company or chain, and typically extends to the lowest level, the store. For example, a three-level location hierarchy might consist of Company, Region, and Store. Each entry (row) in the location hierarchy standard interface describes a specific location. In the example of a location hierarchy shown in Table 3–9, " Location Hierarchy Sample Data", each record describes the region and company of a specific store.

Data Fields

The location hierarchy can have up to twelve levels. Each level in the location hierarchy, just like the merchandise hierarchy, is described by three fields:

- HIERARCHY_ID – an identifier or value for the hierarchy level that is meaningful to the end user. It does not have to be unique.
- HIERARCHY_KEY – a key used to identify the location level that is unique across the chain for that level. It is used to reference the location in other data files.
- HIERARCHY_DESC – a description for the level that describes that level in the location hierarchy.

These three fields are required for each level of the location hierarchy that is used. For example, if a retailer's location hierarchy contains three levels, then the location hierarchy file will contain nine required fields. Any unused fields in the location hierarchy file should be present in the file as NULL (that is, consecutive delimiters) when the file is sent in delimited file format.

An Example

The following table shows sample data for a three-level location hierarchy that consists of Company, Region, and Store.

Table 3–9 Location Hierarchy Sample Data

Hierarchy 1 (Company)			Hierarchy 2 (Region)			Hierarchy 3 (Store)		
ID	**Key**	**Desc**	**ID**	**Key**	**Desc**	**ID**	**Key**	**Desc**
1	1	Full Line	1	FL1	Northeast	1000	1000	New York
1	1	Full Line	2	FL2	Southeast	1001	1001	Atlanta
1	1	Full Line	2	FL2	Southeast	1010	1010	Charlotte
1	1	Full Line	3	FL3	Resort	1002	1002	Puerto Rico
2	2	Outlet	1	O1	Northeast	2000	2000	Philadelphia
2	2	Outlet	2	O2	Southeast	1003	1003	Atlanta

Technical Notes

The following list provides details to consider regarding the location hierarchy data.

- The best way to create a unique Key for each level in the location hierarchy depends on the retailer's hierarchy data. Whenever possible, the hierarchy Keys should not be dependent on higher levels in the hierarchy. In this way, Promote can automatically detect and handle hierarchy moves without additional data. For more information on how Promote manages location hierarchy changes, see "Location Hierarchy Rename Standard Interface Description" on page 3-15.
- The location hierarchy file must contain a record for each location that is referenced in any of a given week's data files.
- The location hierarchy must be described consistently throughout the data file: each hierarchy node must have the same hierarchy ancestors for all records in the file that describes the hierarchy node. In the example shown in Table 3–9 on page 13, the two records describing the hierarchy above Region FL2 are identical. Note that this consistency requirement applies to all three of the hierarchy fields (Key, ID, and Desc). Inconsistent values for hierarchy descriptions are a common reason why some location hierarchy records fail to load.
- Each node in a hierarchy can only have one parent node.
- The lowest level in the location hierarchy should be the level at which sales data is provided.
- The historical location hierarchy should contain a record for each location that is referenced in any historical sales records, even if the location is now closed. It is recommended that retailers provide a single location hierarchy file for all the historical data, rather than one file for each historical week.

Location Hierarchy Attributes Standard Interface

The Location Hierarchy Attributes interface provides store-level attributes for stores the have been defined in the Location Hierarchy standard interface. It also provides the definition and attributes for distribution centers. Promote uses the information to create store sets and to view history by store attributes. The values for OPEN_DT and CLOSE_DT can be provided in advance so that Promote can include either new stores or exclude stores that are closing from the upcoming planning.

Data Feeds

Twenty-eight fields describe a location hierarchy attribute record:

- Location_Key – Unique identifier for location hierarchy.
- LOCATION_LEVEL – Level within location hierarchy.
- MARKET_NAME – Market name.
- STORE_CITY – City.
- STORE_STATE – State.
- LOCATION_TYPE – Store class.
- STORE_NAME – Store name.
- STORE_POSTAL_CODE – Store postal code.
- NSLS_SQF T – Net square footage.
- GRSS_SQFT – Gross square footage.

- OPEN_DT – Beginning of promotion.
- CLOSE_DT – End of promotion.
- CLIMATE – Climate code.
- STORE_FASHION_SEGMENT – Fashion segment code.
- STORE_AD_GROUP – Ad designation.
- STORE_SSC – Store service center (DC) number.
- STORE_CLSS_IND – Store class size.
- SSC_IND – Store service center indicator.
- STORE_CHST_1 – Store characteristic 1.
- STORE_CHST_2 – Store characteristic 2.
- STORE_CHST_3 – Store characteristic 3.
- PRICING_GROUP – Pricing group.
- COMBO_STORE – Combo store.
- TAXABILITY – Taxability.
- STORE_ZIP – Store zip code.
- VOLUME_GR – Gross...
- STORE_CLASS – Store class.
- GRS_ARE_SQFT – Gross square area.

Location Hierarchy CDA Standard Interface Description

The location hierarchy cda interface provides 24 additional optional attributes.

Location Hierarchy Rename Standard Interface Description

The location hierarchy rename interface facilitates moving locations within the location hierarchy. You can rename any node in the hierarchy by supplying the old node name, the new node name, and the level in the hierarchy. You cannot do this through the Location Hierarchy Standard Interface.

Merchandise Hierarchy Standard Interface Description

The merchandise hierarchy interface describes how a retailer categorizes merchandise. The merchandise hierarchy begins with the highest level, such as company or division, and typically extends to the style-color level. For example, a five-level merchandise hierarchy might consist of Division, Department, Class, Style, and Color. Each entry (row) in the merchandise hierarchy standard interface describes the hierarchy for a specific piece of merchandise. In the example of a merchandise hierarchy shown in Table 3–10 on page 16, the merchandise is an item of a specific color, and each row in the file describes the Division, Department, Class, and Style to which the specific color belongs.

Data Fields

The merchandise hierarchy can have up to fifteen levels. Each level in the merchandise hierarchy is described by three fields:

- HIERARCHY_ID – an identifier or value for the hierarchy level that is meaningful to the end user. It does not have to be unique.
- HIERARCHY_KEY – a key used to identify the merchandise level that is unique across the chain for that level. It is used to reference the merchandise in other data files.
- HIERARCHY_DESC – a description for the level that describes that level in the merchandise hierarchy.

These three fields are required for each level of the merchandise hierarchy that is used. For example, if a retailer's merchandise hierarchy contains five levels, then the merchandise hierarchy file will contain fifteen required fields. Any unused fields in the merchandise hierarchy file should be present in the file as NULL (that is, consecutive delimiters) when the file is sent in delimited file format.

Note that the weekly load process expects the merchandise hierarchy to remain the same. It tries to reconcile changes between the new data feed and the existing data by comparing the client_load_id and the level of each record.

- If a particular client_load_id (at a certain level) is present in the feed, but not in the target database, then the node/SKU is added.
- If a particular client_load_id (at a certain level) is present in the feed and in the target database, then the node/SKU is updated, if necessary.
- If a particular client_load_id (at a certain level) is present in the target database but is not in the feed, then the node/SKU is de-activated.

An Example

The following table shows sample data for a five-level hierarchy that consists of Division, Department, Class, Style, and Color. (The hierarchy descriptions are not included here).

Table 3–10 Merchandise Hierarchy Sample Data

Hierarchy 1 (Division)		Hierarchy 2 (Dept.)		Hierarchy 3 (Class)		Hierarchy 4 (Style)		Hierarchy 5 (Color)	
ID	**Key**	**ID**	**Key**	**ID**	**Key**	**ID**	**Key**	**ID**	**Key**
1	1	10	10	20	1020	1234	101234	9	101234509
1	1	10	10	20	1020	1234	101234	12	101234512
6	6	60	60	20	6020	1234	601234	12	601234512

In this example, the class, style, and color levels all have ID values that are not unique across the chain. Because of this, the Key values for these three levels cannot be the same as the ID values. The unique Key values for these three levels were created by combining values from higher levels in the hierarchy. The Key for the Class level was created by appending the Class ID to the Department Key. The Key for the Style level was created by appending the Style ID to the Department Key.

Technical Notes

The following list provides details to consider regarding the merchandise hierarchy data.

- The best way to create a unique Key for each level in the merchandise hierarchy depends on the retailer's hierarchy data. Whenever possible, the hierarchy Keys should not be dependent on higher levels in the hierarchy. In this

way, Promote can automatically detect and handle hierarchy moves without additional data. For more information on how Promote manages merchandise hierarchy changes, see "Merchandise Hierarchy Rename Standard Interface Description" on page 3-19.

- The merchandise hierarchy file must contain a record for each product that is referenced in any other of a given week's data files.
- The merchandise hierarchy must be described consistently throughout the data file: each hierarchy node must have the same hierarchy ancestors for all records in the file that describes the hierarchy node. In the example shown in Table 3–10 on page 16, the first two records describe the hierarchy above Style 101234 in an identical way. Note that this consistency requirement applies to all three of the hierarchy fields (Key, ID, and Desc). Inconsistent values for hierarchy descriptions are a common reason why some merchandise hierarchy records fail to load.
- Each node in a hierarchy can only have one parent node.
- The lowest level in the merchandise hierarchy must be the level at which sales and distribution data are provided.
- The historical data files should include a record for each product that is referenced in any historical sales records, even if the product is inactive. It is recommended that retailers provide a single merchandise hierarchy file for all the historical data, rather than one file for each historical week.

Merchandise Hierarchy Attributes Standard Interface Description

The merchandise hierarchy attributes interface provides information about merchandise attributes at various levels in the MH, principally lot/color/line/sku. This information is used to provide context for merchandise during planning and allocation.

Data Feeds

Forty-four fields describe a merchandise hierarchy attribute record:

- MERCHANDISE_KEY – the unique identifier for the merchandise hierarchy.
- MERCHANDISE_LEVEL – the level within the merchandise hierarchy.
- BRAND – the ID of the brand.
- BRAND_DESC – the description of the brand.
- VENDOR – the number of the supplier. This field contains the manufacturer number when the supplier is set as a warehouse.
- VENDOR_DESC – the description of the supplier.
- ITEM_SIZE – the physical size of the item.
- CATEGORY - the category.
- CATEGORY_DESC – the description of the category.
- REPORT_CLIENT_ID – the client ID associated with the report.
- START_DT – the date specifying the beginning of the plan.
- FIRST_CREATE_DT – the date when the merchandise was first introduced.
- LAST_MODIFIED_DT – the time stamp of the last modification.
- PROD_LEVEL – the product level.

- COST – the wholesale cost.
- RETAIL – the retail price.
- PACK_SIZE – the pack size (inner).
- SIZE_RANGE_DESC – the description of the size range.
- DISP_CODE – the disposition code.
- PURCH_TYPE – values are Basic (B), Fashion (F), and Key (K).
- GRP_IN – the group indicator.
- PROD_TYPE – the product type.
- BRAND_NAME – the brand name.
- CNTL_RKL – the control rkl.
- COLL_ID – the ID of the collection.
- COLL_NAME – the name of the collection.
- MSTR_COLL_IND – the master collection indicator.
- ORIG_IND – the indicator for the origin (Domestic or Import).
- WEIGHT – the weight.
- COLOR_CNT – the number of colors per style.
- SIZE_GRP_DESC – the description of the size group.
- LINE_PCT – the line percent.
- OOS_DATE – the season out-of-stock date.
- VENDOR_STYLE – the vendor style number.
- ALLOC_FLAG – the allocate flag (RAP)
- FIRST_EFF_DT – the date when the merchandise is first in effect. Prior to this date, the merchandise will behave as if it is excluded. This date must be earlier than LAST_EFF_DT.
- LAST_EFF_DT – the last date when the merchandise is in effect. After this date, the merchandise will behave as if it is excluded. This date must be later than FIRST_EFF_DT.
- BRAND_TYPE – not used.
- PROMO_EXCLUSION – used to indicate that a record is excluded (Y) or not (N). Excluded records still appear in the UI, both in the MH browser tree and the in the Promotion Offer SKU view. The records in the SKU view will be flagged as excluded and will not be forecasted or used in metrics.
- MERCHANDISE_SUBTYPE – the season code.
- SIZE_RANGE_KEY – the ID of the size range.
- SIZE_KEY – the size ID.
- MERCHANDISE_FLOOR_SET – the subset of a season that defines when an item is introduced to the floor.
- COLOR_FAMILY – the color family.

Merchandise Hierarchy CDA Standard Interface Description

The merchandise hierarchy cda interface provides 24 additional optional attributes.

Merchandise Hierarchy Rename Standard Interface Description

The merchandise hierarchy rename interface facilitates moving merchandise within the merchandise hierarchy. Pieces of the merchandise hierarchy can be moved while renaming the client_load_ids of nodes (with the exception of SKUs) within the same level. For example, a subclass can be moved from class A to class B and the client_load_id of that subclass is changed. Any node in the hierarchy can be renamed by supplying the old node name, the new node name, and the level in the hierarchy. This cannot be done through the Merchandise Hierarchy Standard Interface. This change must be part of the Merchandise Hierarchy load as well. Note that the MH Rename load must be completed before the MH load for the change to occur.

Technical Notes

Note that this information pertains to both the Merchandise Hierarchy Rename Standard Interface and the Location Hierarchy Rename Standard Interface.

The application database associates other information with a node in the merchandise (or location) hierarchy through an internally generated key. Each node of the hierarchy has one of these internal keys in addition to the key that is sent by a client. Information like historical sales records, analytical parameters, and business rules is stored according to these internal keys. The relation between the internal keys and the client keys must be preserved when hierarchies are changed.

The rename interface is used to update the association between the client key and an internal key after a re-class occurs. The association between the client key and the internal key is updated by specifying the old key, the new key, and the level. The rename interface always needs to be combined with a merchandise hierarchy reflecting the changes that have been made. In the most general case, both of these files are required to fully specify a hierarchy change.

It is recommended that the keys at each level of the hierarchy should be unique without depending on parent levels so that hierarchy changes can be made without sending a rename file. In that case, when a node is moved, the changed hierarchy is sent. Since the keys for the nodes that move are unchanged, the internal keys will retain the correct association and nothing else needs to happen. The new parent-child relationships are simply defined by the latest hierarchy.

It may not be practical to provide keys at all levels that are independent of the keys at the parent level. For example, the CLASS key concatenates the DEPT and CO keys above it. This implies that the rename interface is needed for certain types of hierarchy changes, as discussed below.

Another important concept is that the rename interface can be used for a "move" in the merchandise hierarchy, but does not directly describe a "merge". So, for example, there is no direct way to specify (assuming Dept 42 already exists):

"Move Department 44 to Department 42"

However, the desired result can be accomplished by:

"Move all classes in Department 44 into Department 42"

The types of moves specified below fall into the following categories:

- Move all departments in one division into another division.
- Move all classes in one department to another department.

- Move some classes from one department to another department.

The way to accomplish these moves depends on how the keys, at and below the levels in question, will be affected.

Case 1 When departments are moved to another division, the keys at and below department will not change, since division is not incorporated in the key. (The exception would be if a division were moved into another company.) Since the keys do not change at department or below, this move can be accomplished by sending the new merchandise hierarchy, with departments that were in the old division having the new division as their parent.

Cases 2 and 3 When classes are moved to another department, the keys at and below class for the affected nodes will all change (since class keys and below are all constructed by concatenating the class into the key). In these cases, a rename file must be sent in addition to the updated merchandise hierarchy. This file will contain a record for the affected class and for each of its descendents.

For example, in order to move CLASS 0263 from DEP 0059 to DEP 0086 (the class has STYLEs 0001 and 0002, each with HALF-SIZEs 0 and 1, each with COLORs 0001 and 0002), the following records in the rename file must be sent:

```
TOO00590263|TOO00860263|CLASS
TOO005902630001|TOO008602630001|STYLE
TOO005902630002|TOO008602630002|STYLE
TOO005902630010|TOO008602630010|HALF-SIZE
TOO005902630011|TOO008602630011|HALF-SIZE
TOO005902630020|TOO008602630020|HALF-SIZE
TOO005902630021|TOO008602630021|HALF-SIZE
TOO0059026300100001|TOO0086026300100001|COLOR
TOO0059026300110001|TOO0086026300110001|COLOR
TOO0059026300200001|TOO0086026300200001|COLOR
TOO0059026300210001|TOO0086026300210001|COLOR
TOO0059026300100002|TOO0086026300100002|COLOR
TOO0059026300110002|TOO0086026300110002|COLOR
TOO0059026300200002|TOO0086026300200002|COLOR
TOO0059026300210002|TOO0086026300210002|COLOR
```

These records tell the application how to associate the internal keys at each node with the new keys. (The new merchandise hierarchy file should also reflect the result of the moves.)

Merchandise Thresholds Standard Interface Description

This interface defines thresholds for merchandise hierarchy levels. The threshold number represent the minimum percent that is required to meet that threshold. Green must have a higher value than Yellow. Yellow must have a higher value than Red. The values should be provided for the CHAIN merchandise level. In addition, any merchandise level that should have a different threshold should also be provided.

Data Fields

Five fields describe a merchandise thresholds record:

- MERCH_CLIENT_LOAD_ID – the client ID.
- MERCH_LEVEL_DESC – the description provided can be at any level that is equal to or higher than the PROMOTE_MIN_LCD configuration.

- GREEN_THRESHOLD – high confidence in the accuracy of the forecast.
- YELLOW_THRESHOLD – medium confidence in the accuracy of the forecast.
- RED_THRESHOLD – low confidence in the accuracy of the forecast.

An Example

The following is an example of the data for merchandise thresholds.

Table 3–11 Merchandise Threshold Example

Merch Client Load ID	Merch Level Desc	Green Threshold	Yellow Threshold	Red Threshold
1	CHAIN	70	30	0
2	DEPARTMENT	60	40	0

Offers Standard Interface Description

The offers interface contains the master data that describes a client's specific promotion (for example, a 2 for 1 promotion).

Data Fields

Nine fields describe an offer:

- NAME – the display name for the offer.
- INACTIVE – activity flag. A value of 0 indicates the offer is active; a value of 1 indicates the offer is inactive.
- EXTERNAL_NAME – the ID for the offer that is meaningful to the client. It is unique across all offers.
- DESCRIPTION – an optional description of the offer.
- BUSINESS_RULE_CLASS_NAME – the instance of what class to use in the validation.
- TYPE_EXTERNAL_NAME – the name of the user-defined type.
- MODEL_CODE – the bit identifier of the offer. The value must be a power of 2 and is unique across the universe of all offers (for example, 0, 1, 2, 4, 8...).
- FORMAT – the output format for the offer (for example to put $in front of the number).
- TYPE_ENUM – 0 = integer; 1 = user-defined; 2 = decimal; 6 = none.

An Example

The following is an example of the data for an offers record.

Table 3–12 Offers Example Data

Name	Inactive	External Name	Descrip-tion	Business Rule Class Name	Type External Name	Model Code	Format	Type Enum
% Off	0	offer.percent_off	% Off	com.profitlogic.promote.bean.rule.PercentOffOfferRule	ude.percent.off	1	{0}	0

Period Attributes Standard Interface Description

The period attributes interface is used to indicate whether or not there are active promotions for given periods.

Four fields describe period attributes:

- BEGIN_CALENDAR_DT – beginning of the period.
- END_CALENDAR_DT – end of the period.
- DARKPERIOD_FLAG – defines whether or not there are active promotions during the specified dates.
- DARKPERIOD_DESC – provides an optional description of the dark period.

Promotion Allocation Standard Interface Description

The promotion allocation interface provides a way to import historical space allocation usage. This applies only to promotions managed external to the application.

Data Fields

Four fields describe a promotion allocation:

- PROMO_EXTERNAL_NAME – the ID for the promotion that is meaningful to the client.
- MERCH_CLIENT_LOAD_ID – the client-specific category ID.
- LEVEL_DESC – the client-specific merchandise hierarchy level description.
- SPACE_ALLOCATION – the allocation for the given category.

An Example

The following is an example of the data for a promotion allocation.

Table 3–13 Promotion Allocation Example Data

Promo External Name	Merch Client Load ID	Level Desc	Space Allocation
1-003-1-999000002	236	DEPARTMENT	0.1

Promotion Campaign Standard Interface Description

The promotion campaign interface describes a client's promotional data. This data feed provides Promote with promotional calendar information from other systems. It is also used to import historical data into the system for ad effectiveness analysis.

Data Fields

Six fields describe a promotion campaign.

- NAME – a display name for the campaign.
- DESCRIPTION – an optional description of the campaign.
- EXTERNAL_NAME – the ID for the campaign that is meaningful to the client. It is unique across all campaigns.
- BEGIN_DATE – the start date for the campaign.
- END_DATE – the end date for the campaign.
- INACTIVE – activity flag. A value of 0 indicates the campaign is active; a value of 1 indicates the campaign is inactive.

An Example

The following is an example of the data for a promotion campaign.

Table 3–14 Promotion Campaign Example Data

Name	Description	External Name	Begin Date	End Date	Inactive
campaign0001	BTS Campaign	Campaign for Back-to-School	2003-10-10	2003-10-17	1

Promotion Forecaster Standard Interface Description

The promotion forecaster interface is used for to allow an external source to specify the tasks that must be done by the promotion forecaster agent. This agent reads the target table in order to get the required tasks and runs them in the sequence specified (if provided). This allows an external system to define the priorities as to what should be forecasted and when.

Note that all of the columns in the interface are technically nullable; FORECAST, FORCE< and REFRESH are best not left Nullable. The loader has validation rules to ensure that at least one column has been provided. Either a PromoExternalName or both PromoExternalName & OfferExternalName or EventExternalName must be provided. If an EventExternalName is provided, then the loader will expand this out so that all promotions are processed.

Data Fields

Seven fields describe a promotion forecaster record:

- PROMO_EXTERNAL_NAME – the ID for the promotion that is meaningful to the client.
- OFFER_EXTERNAL_NAME – the ID for the offer that is meaningful to the client. It is unique across all offers.
- ORDER_ID – Order ID.

- EVENT_EXTERNAL_NAME – the name of the event used for the promotion.
- FORECAST – Flag that indicates whether or not to forecast a given promo/offer. 1 = forecast. 0 = skip. Although this attribute is technically nullable, it is recommended that it not be left nullable.
- FORCE – Flag that indicates whether or not to force the forecast of a given promo/offer. 1 = force. 0 = skip. Although this attribute is technically nullable, it is recommended that it not be left nullable.
- REFRESH –Flag that indicates whether or not to refresh a given offer's criteria. 1 = refresh. 0 = skip. Although this attribute is technically nullable, it is recommended that it not be left nullable.

Promotion Offer Standard Interface Description

The promotion offer interface describes all the offers in a promotion.

Data Fields

Fourteen fields describe a promotion offer:

- INACTIVE – the status of a promotion offer. A value of 0 indicates that the promotion offer is active. A value of 1 indicates that the promotion offer has been deleted. The default is active.
- NAME – the display name for the offer.
- EXTERNAL_NAME – the ID for the offer that is meaningful to the client.
- DESCRIPTION – an optional description of the offer.
- BEGIN_DATE – the start date for the offer.
- END_DATE – the end date for the offer.
- PROMO_EXTERNAL_NAME – the ID for the promotion that is meaningful to the client.
- OFFER_EXTERNAL_NAME – the ID for the offer that is meaningful to the client. It is unique across all offers.
- EVENT_EXTERNAL_NAME – an identifier that associates an offer with other offers in the same event.
- UDV_EXTERNAL_NAME – the actual user-defined type value.
- VALUE_INT – the integer value of the offer (either UDV_EXTERNAL_ NAME,VALUE_INT, or VALUE_DEC should be set).
- VALUE_DEC – the decimal value for the actual offer.
- PAGE_NUM – the page of the offer.
- POS_NUM – the position of the offer.

An Example

The following is an example of the data for a promotion offer.

Table 3–15 Promotion Offer Example

Inac-tive	Name	Exter-nal Name	Descr ip tion	Begin Date	End Date	Promo Exter-nal Name	Offer Exter-nal Name	Event Exter_ nal Name	UDV Exter-nal Name	Value Int	Value Dec	Page Num	Pos Num
0	Barbie Sale	7293	CG Barbie Offer	2003-01-31	2003-02-02	1-001-1-9010	offer.per cent.off	P1B1S1	udev.per cent_off.10			1	2

Promotion Offer Attributes Standard Interface Description

The promotion offer attributes interface describes the additional attributes for each offer (for example, page position: front, middle, and back).

Data Fields

Six fields describe a promotion offer attribute.

- PROMO_EXTERNAL_NAME – the ID for the promotion that is meaningful to the client.
- PROMO_OFFER_EXTERNAL_NAME – the ID for the promotion offer that is meaningful to the client. It is unique across all promotion offers.
- VEH_ATTR_EXTERNAL_NAME – the vehicle attribute name that is meaningful to the client.
- UDV_EXTERNAL_NAME – the actual user-defined type value.
- VALUE_INT – the integer value of the offer (either UDV_EXTERNAL_NAME,VALUE_INT, or VALUE_DEC should be set).
- VALUE_DEC – the currency value for the actual offer.

An Example

The following is an example of the data for a promotion offer attribute.

Table 3–16 Promotion Offer Attribute Example Data

Promo External Name	Promo Offer External Name	Veh Attr External Name	UDV External Name	Value Int	Value Dec
1-001-1-999000000	LR-9990000000 T0000099958	page_location	udev.page_location.front		

Promotion Offer Criteria Standard Interface Description

The promotion offer criteria interface describes the definition of items set to be included or excluded from a promotion offer.

Data Fields

Fifteen fields describe a promotion offer criteria record.

- INACTIVE – the activity flag. A value of 0 indicates that the SKU list is active. A value of 1 indicates that the SKU list has been deleted. The default is active.
- EXTERNAL_NAME – the ID of the SKU list. It is meaningful to the client and is unique across SKU lists.
- PROMO_EXTERNAL_NAME – an ID, which is meaningful to the client, for the promotion specific to this offer criterion.
- PROMO_OFFER_EXTERNAL_NAME – an ID, which is meaningful to the client, for the promotion offer specific to this offer criterion.
- CRITERION_TYPE – the type of offer criterion. A value of 0 indicates a type of SKU list (identified by SKU_LIST_EXTERNAL_NAME). A value of 1 indicates a type of merchandise category – for example, class or subclass – (identified by MERCH_CLIENT_LOAD_ID and LEVEL_DESC). A value of 2 indicates SKU (identified by MERCH_CLIENT_LOAD_ID).
- SKU_LIST_EXTERNAL_NAME – if CRITERION_TYPE = 0, then this is used to provide a meaningful identifier.
- MERCH_CLIENT_LOAD_ID – if CRITERION_TYPE = 1, then this is used to provide a meaningful identifier.
- LEVEL_DESC – the level of the category.
- ATTRIBUTE_NAME – this value restricts the criterion type. Supported values are RETAIL and VENDORNAME, which can be found in the Merchandise Hierarchy.
- ATTRIBUTE_VALUE – restricts the category (for CRITERION_TYPE 1) by this attribute value.
- ATTRIBUTE_NAME2 – this value restricts the criterion type. Supported values are RETAIL and VENDORNAME, which can be found in the Merchandise Hierarchy.
- ATTRIBUTE_VALUE2 – restricts the category (for CRITERION_TYPE 1) by this attribute value.
- LOGICAL_OPERATOR – indicates how the two category filters (ATTRIBUTE_NAME / ATTRIBUTE_VALUE and ATTRIBUTE_NAME2 / ATTRIBUTE_VALUE2) are combined. Values are 0 (= or) and the default value of 1 (= and).
- INCLUDE – a value of 1 indicates that the SKUs specified by this criterion are included. A value of 0 indicates that the SKUs are excluded.
- LIST_TYPE – Buy List = 0; Get List = 1.

An Example

The following is an example of the data for a promotion offer criteria record, with some optional attributes omitted.

Table 3–17 Promotion Offer Criteria Example Data

Inactive	External Name	Promo External Name	Promo Offer External Name	Criterion Type	SKU List External Name	Logical Operator	Include	List Type
0	All Flavors	Promo_1	Promo Offer 1	0	List_1	1	0	1

Promotion Offer Merchandise Standard Interface Description

The promotion offer merchandise interface describes the SKUs associated with an offer.

Data Fields

Seven fields describe a promotion offer merchandise record.

- PROMO_EXTERNAL_NAME – the ID for the promotion that is meaningful to the client.
- PROMO_OFFER_EXTERNAL_NAME – the ID for the promotion offer that is meaningful to the client. It is unique across all promotion offers.
- MERCH_CLIENT_LOAD_ID – the client-specific category ID.
- LEVEL_DESC – the client-specific merchandise hierarchy level description.
- FULL_PRICE – the price of the item.
- PROMO_PRICE – the promotion price of the item.
- COST – the actual cost of the item.

An Example

The following is an example of the data for a promotion offer merchandise record.

Table 3–18 Promotion Offer Merchandise Example Data

Promo External Name	Promo Offer External Name	Merch Client Load ID	Level Desc	Full Price	Promo Price	Cost
1-001-1-999000000	LR-999000000-0 T0000099958	T0000099958	SKU	24.50	18.37	12.25

Promotion Offer Store Standard Interface Description

The promotion offer store interface describes the stores on a promotion.

Data Fields

Three fields describe a promotion offer stores record.

- PROMO_EXTERNAL_NAME – the ID for the promotion that is meaningful to the client.
- LOC_CLIENT_LOAD_ID – the client-specific store ID.
- LEVEL_DESC - the client-specific store hierarchy level description.

An Example

The following is an example of the data for a promotion offer store record.

Table 3–19 Promotion Offer Store Example Data

Promo External Name	Loc Client Load ID	Level Desc
1-001-1-999000000	6493	STORE

Promotions Standard Interface Description

The promotions interface describes a client's promotions data. The data feed provides Promote with promotional calendar information from other systems. It is also used to import historical data into the system that is used for ad effectiveness analysis.

Data Fields

Twelve fields describe a promotion record.

- TYPE – defines the promotion. A value of 4 indicates an historical promotion. A value of 5 indicates a pre-planned promotion.
- INACTIVE – the status of a promotion. A value of 0 indicates that the promotion is active. A value of 1 indicates that the promotion is inactive. The default is 0. This attribute is required as part of the data feed. If the value is inactive, then the record will not be displayed in the UI.
- NAME – the display name for the promotion.
- EXTERNAL_NAME – the ID for the promotion that is meaningful to the client. It is unique across all promotions.
- DESCRIPTION – an optional description of the promotion.
- BEGIN_DATE – the start date for the promotion.
- END_DATE – the end date for the promotion.
- TOTAL_COST – the total cost allocated to the promotion.
- VEHICLE_EXTERNAL_NAME – the vehicle that is used when promoting items.
- PAGES – the number of pages for the vehicle.
- EVENT_EXTERNAL_NAME – the name of the event used for the promotion.
- CAMPAIGN_EXTERNAL_NAME – the name of the campaign being used when promoting items.

An Example

The following is an example of the data for a promotion record.

Table 3–20 Promotion Example Data

Type	Inactive	Name	External Name	Description	Begin Date	End Date	Total Cost	Vehicle External Name	Pages	Event External Name	Campaign External Name
4	0	Circular for Week 20	promo 0001	Standard Weekly Circular	2003-10-10	2003-10-17	120000.00	vehicle.circular	4		Campaign for Back-to-School

Seasonal Trend Standard Interface Description

The seasonal trend interface describes the adjustment to the seasonal SKU based on a comparison between historical data and current data.

Data Fields

Five fields describe a seasonal trend record.

- BEGIN_CALENDAR_DT – the beginning date for the data being analyzed.
- END_CALENDAR_DT – the end date for the data being analyzed.
- LOC_CLIENT_LOAD_ID – the external ID for the location.
- LOC_LEVEL_DESC – the client-specific location level description
- USR_TREND – the value applied to the seasonal forecast.

Seasonalities Standard Interface

The seasonalities standard interface describes the seasonality values (effects related to the time of year) provided by Analytical Services that are used by Promote for calculations.

Data Fields

Eight fields describe a seasonality map record:

- PRIORITY – the search priority for the seasonality.
- SEASONALITY_ID – the ID for the seasonality.
- MERCHANDISE_LEVEL – description of the level of the merchandise hierarchy.
- MERCHANDISE_KEY – key for the merchandise hierarchy level.
- LOCATION_LEVEL – description of the level of the location hierarchy.
- LOCATION_KEY – key for the location hierarchy level.
- ATTRIBUTE_VALUE_MASK – the search mask that specifies the season code and, optionally, the item attributes of the seasonality curves.
- AS_VERSION – the version number for the current run. Set by Analytical Parameter Calculator (APC) and used to track run versions.

Six fields describe a seasonality values record:

- SEASONALITY_ID – the ID for the seasonality.
- CALENDAR_DT – the date for the seasonality.
- SEAS_INDX – the value for the seasonality for the date.
- SEAS_ERR – for future use. Set to 0.
- AS_PARAMETER_ID – a number that uniquely identifies the current record and that is used for tracking.
- AS_VERSION – the version number for the current run. Set by APC and used to track run versions.

SKU List Standard Interface Description

The SKU list interface describes the list of SKUs for a promotion.

Data Fields

Four fields describe a SKU list record.

- INACTIVE – the activity flag. A value of 0 indicates that the SKU list is active. A value of 1 indicates that the SKU list has been deleted. The default is active.
- NAME – the display name for the SKU list.
- EXTERNAL_NAME – the ID for the SKU list that is meaningful to the client. It is unique across all SKU lists.
- DESCRIPTION – an optional description of the SKU list.

An Example

The following is an example of the data for a SKU list record.

Table 3–21 SKU List Example Data

Inactive	Name	External Name	Description
0	Crafted Bouquet	897	Double Leaf and Long Leaf

SKU List Items Standard Interface Description

The SKU list items interface describes the items on a SKU list for a promotion.

Data Fields

Two fields describe a SKU list item record.

- SKU_LIST_EXTERNAL_NAME – the ID of the parent SKU list.
- MERCH_CLIENT_LOAD_ID – the customer's Like Merchandise ID.

An Example

The following is an example of the data for a SKU list item record.

Table 3–22 SKU List Item Example Data

SKU List External Name	Merch Client Load ID
897	T0000015167

Store Set Price Standard Interface Description

The store set price interface is used for the versioning of a promotion. This interface permits the storing of zone level pricing data. This is made possible by having the client provide Store Sets and Store Subsets that define the different price zones used by the client. This can be handled via the existing Store Set, Subset and Assignment interfaces.

Data Fields

Six fields describe a store set price record:

- MERCH_CLIENT_LOAD_ID – the customer's ID.
- MERCH_LEVEL_DESC – description of the merchandise level.
- STORE_SET_NAME – the name for the store set.
- STORE_SUBSET_NAME – the name for store subset.
- PRICE - the price for the version of the promotion.
- COST – the associated cost.

Store Sets Standard Interface Description

The store sets interface describes a client's store set configuration. This standard interface provides fields that can be used to change the name of the store set and to assign a name to the remaining subset. Note that OLD_STORE_SET_NAME and REMAIN_SUBSET_NAME are optional.

Data Fields

Eight fields describe a store set record.

- NEW_STORE_SET_NAME – the new name for the store set, which is assigned if a value is provided for the old store set name attribute.
- OLD_STORE_SET_NAME – the existing name for the store set. This attribute is optional. If a value is provided, then the existing name for the store set is replaced by the new name that is provided in NEW_STORE_SET_NAME.
- STORE_SET_DESC – the description of the store set.
- INACTIVE – a flag to indicate the status of the store set. 1 = active; 0 = inactive.
- STORE_SET_TYPE – 0
- FIRST_EFF_DT – the date when the store set becomes active.
- LAST_MODIFIED_DATE – the date when the record was last modified.
- REMAIN_SUBSET_NAME – the name for the remaining subset for the associated store set. This value is optional. The naming occurs only if a value is provided for this attribute. (A remaining store subset is a special type of subset. Only one is allowed, and it contains all unassigned subsets.)

An Example

The following is an example of the data for a store set record.

Table 3–23 Store Sets Example Data

New Store Set Name	Old Store Set Name	Store Set Description	Inactive	Store Set Type	First Effective Date	Last Modified Date	Remaining Subset Name
New Name	Existing Name	Default system store set	1	0	2006-02-14	2006-02-14	Subset Remain Name

Store Subsets Standard Interface Description

The store subsets interface describes a client's store subset configuration. This standard interface provides fields that can be used to change the name of the store subset. Note that OLD_STORE_SUBSET_NAME is optional.

Data Fields

Six fields describe a store subset record.

- NEW_STORE_SUBSET_NAME – the new name for the store subset, which is assigned if a value is provided for the old store subset name attribute.
- OLD_STORE_SUBSET_NAME – the existing name for the store subset. This attribute is optional. If a value is provided, then the existing name for the store subset is replaced by the new name that is provided in NEW_STORE_SUBSET_NAME.
- STORE_SUBSET_DESC – the description of the store subset.
- STORE_SET_NAME – the name of the store set related to this store subset.
- INACTIVE – a flag to indicate the status of the store subset. 1 = active; 0 = inactive.
- ORDER_SEQ – the position of the subset.

An Example

The following is an example of the data for a store subset record.

Table 3–24 Store Subset Example Data

New Store Subset Name	Old Store Subset Name	Store Subset Description	Store Set Name	Inactive	Order Sequence
New Name	Northeast	Northeast subset	Default	1	0

Store Subset Assignments Standard Interface Description

The store subset assignments interface describes a client's store subset assignments.

Data Fields

Four fields describe a store subset assignment record.

- LOC_CLIENT_LOAD_ID – the external ID for the location.
- LEVEL_DESC – the external ID for the location level.
- STORE_SUBSET_NAME – the name of the store subset for the location.
- STORE_SET_NAME – the store set name for the location.

An Example

The following is an example of the data for a store subset assignment record.

Table 3–25 Store Subset Assignment Example Data

Location Client Load ID	Level Description	Store Subset Name	Store Set Name
5169	Store	Central	Default

TAE Temp Metric Standard Interface Description

The TAE temp metric interface describes the data loaded into a temporary table for use in reporting and comparison.

Data Fields

Forty four fields describe a TAE temp metric record:

- RUN_ID – the execution ID.
- PROMO_ID – the internal promotion ID.
- AD_DATE – the date of the promotion.
- PI_ID – the merchandise ID.
- LOCATION_ID – the internal location ID.
- AD_ITEM_PRICE – TAE-generated metric.
- AD_ITEM_ROSALE – TAE-generated metric.
- AD_ITEM_VISIT_RATE – TAE-generated metric.
- AD_ITEM_SALES – TAE-generated metric.
- AD_ITEM_GM – TAE-generated metric.
- TTL_AD_DAYS – TAE-generated metric.
- AD_ITEM_AC_SALES – TAE-generated metric.
- AD_ITEM_AC_GM – TAE-generated metric.
- AD_ITEM_PR_SALES – TAE-generated metric.
- AD_ITEM_PR_GM – TAE-generated metric.
- AD_NONAD_SALES – TAE-generated metric.
- AD_NONAD_GM – TAE-generated metric.
- BL_SUBST_CODE – TAE-generated metric.
- BL_SUBST_ITEM – TAE-generated metric.
- TTL_BASE_PERIODS – TAE-generated metric.
- BL_ITEM_ROSALE – TAE-generated metric.
- BL_ITEM_SALES – TAE-generated metric.
- BL_ITEM_VISIT_RATE – TAE-generated metric.
- BL_ITEM_GM – TAE-generated metric.
- BL_ITEM_PRICE – TAE-generated metric.
- BL_ITEM_AC_SALES – TAE-generated metric.

- BL_ITEM_AC_GM – TAE-generated metric.
- BL_ITEM_PR_SALES – TAE-generated metric.
- BL_ITEM_PR_GM – TAE-generated metric.
- BL_NONAD_SALES – TAE-generated metric.
- BL_NONAD_GM – TAE-generated metric.
- AD_MB_ITEM_ONLY – TAE-generated metric.
- AD_MB_ITEM_AD – TAE-generated metric.
- AD_MB_ITEM_NONAD – TAE-generated metric.
- AD_MB_ITEM_ADNONAD – TAE-generated metric.
- AD_ITEM_OTHAD_ROS – TAE-generated metric.
- STORE_BASE – TAE-generated metric.
- AD_ITEM_NORMAL_PRICE – TAE-generated metric.
- AD_ITEM_AC_UNITS – TAE-generated metric.
- AD_ITEM_PR_UNITS – TAE-generated metric.
- AD_NONAD_UNITS – TAE-generated metric.
- BL_ITEM_AC_UNITS – TAE-generated metric.
- BL_ITEM_PR_UNITS – TAE-generated metric.
- BL_NONAD_UNITS – TAE-generated metric.

Transaction Log Standard Interface Description

The transaction log interface describes a client's basic transactional information. This data feed is used when using Promote's built-in data warehousing feature. Alternative configurations are available when leveraging a client's existing data warehouse.

Data Fields

Eleven fields describe a transaction log record:

- TXN_ID – the unique identifier for the transaction.
- TXN_DATE – the transaction date.
- LOC_CLIENT_LOAD_ID – the ID for the location of the transaction.
- MERCH_CLIENT_LOAD_ID – the ID of the product being sold.
- UNIT_COST – the per-unit cost of the sold product.
- UNIT_NORMAL_PRICE – the per-unit non-promotional price of the sold product.
- UNITS_SOLD – the number of a given item that were purchased in the market basket.
- EXT_RETAIL_AMT – the at-register price for the product being sold.
- EXT_MARGIN_AMT – the amount that the price has been reduced if the item is on promotion for this kind of item in the market basket.
- AD_IND – discount flag. 0 = none; 1 = on Ad (item was promoted); 2 = clearance.
- PROMO_TXN_CODE – an optional field used to capture an offer code, coupon code, or other extended information.

An Example

The following is an example of the data for a transaction log record.

Table 3–26 Values Example Data

Txn ID	Txn	Loc Client Load ID	Merch Client Load ID	Unit Cost	Unit Normal Price	Units Sold	Ext Retail Amt	Ext Margin Amt	Ad Ind	Promo Txn Code
100175	2006-02-28	459901	T8946094	6.0	8.99	2	17.98	6.0	1	C-333

Technical Notes

Transaction log data is partitioned by week in the database. When data for an already-processed week arrives, it is merged into the existing partition. If a substantial (greater than 10 percent) reload is being processed, it may be faster to drop the corresponding partition and re-process the entire week again.

User Defined Type Standard Interface Description

The user defined type interface describes a client-defined type (for example, percent off and page units). The data feed typically provides Promote with user-defined information from other systems.

Data Fields

Four fields describe a user-defined type:

- TYPE_NAME – the display name for the type.
- INACTIVE – activity flag. A value of 0 indicates the offer is active; a value of 1 indicates the offer is inactive.
- EXTERNAL_NAME – the ID for the type that is meaningful to the client. It is unique across all types.
- DESCRIPTION – an optional description of the type.

An Example

The following is an example of the data for a type record.

Table 3–27 Type Example Data

Type Name	Inactive	External Name	Description
% Off	0	ude.percent_off	Percent Off

User Defined Value Standard Interface Description

The user defined value interface describes a value for a client-defined type (for example, 5% for a Percent Off user-defined type). The data feed typically provides Promote with user-defined values information from other systems.

Data Fields

Seven fields describe a user-defined value:

- VALUE_NAME – the display name for the value.
- INACTIVE – activity flag. A value of 0 indicates the offer is active; a value of 1 indicates the offer is inactive.
- EXTERNAL_NAME – the ID for the type that is meaningful to the client. It is unique across all types.
- TYPE_EXTERNAL_NAME – the name of the user-defined type.
- DESCRIPTION – an optional description of the type.
- ORDER_ID – the position of the element in an ordered list.
- EXTERNAL_CODE – the element's ID in the external system.

An Example

The following is an example of the data for a values record.

Table 3–28 Values Example Data

Value Name	Inactive	External Name	Type External Name	Description	Order ID	External Code
10%	0	ude.percent_off.10	ude.percent_off	10% Off	2	1

Vehicle Standard Interface Description

The vehicle interface describes a client's promotion vehicle (for example, circular or TV ad). The data feed typically provides Promote with vehicles information from other systems. It is also used to import historical data into the system for ad effectiveness analysis.

Data Fields

Six fields describe a vehicle:

- VEHICLE_NAME – the display name for the vehicle.
- INACTIVE – activity flag. A value of 0 indicates the offer is active; a value of 1 indicates the offer is inactive.
- EXTERNAL_NAME – the ID for the vehicle that is meaningful to the client. It is unique across all vehicles.
- DESCRIPTION – An optional description of the vehicle.
- BUSINESS_RULE_CLASS_NAME – the instance of what class to use in the validation.
- MODEL_CODE – the bit identifier of the offer. The value must be a power of 2 and unique across the universe of all offers (for example, 0, 1, 2, 4, 8...).

An Example

The following is an example of the data for an vehicles record.

Table 3–29 Vehicles Example Data

Name	Inactive	External Name	Description	Business Rule Class Name	Model Code
Circular	0	vehicle.circular	Circular	com.profitlogic.promote.bean.rule.CircularVehicleRule	1

Vehicle Attributes Standard Interface Description

The vehicle attributes interface describes the attributes of a client's vehicle (for example, pages and space allocation). The data feed typically provides Promote with vehicle attributes information from other systems. It is also used to import historical data into the system for ad effectiveness analysis.

Data Fields

Twelve fields describe a vehicle attribute:

- VEHICLE_ATTR_NAME – the display name for the vehicle attribute.
- INACTIVE – activity flag. A value of 0 indicates the offer is active; a value of 1 indicates the offer is inactive.
- EXTERNAL_NAME – the ID for the vehicle attribute that is meaningful to the client. It is unique across all vehicle attributes.
- DESCRIPTION – an optional description of the vehicle attribute.
- ATTRIBUTE_LEVEL – the level at which to show the attribute. A value of 1 indicates vehicle; a value of 1 indicates item.
- VEHICLE_EXTERNAL_NAME – the ID for the parent vehicle that is meaningful to the client. It is unique across all vehicles.
- TYPE_EXTERNAL_NAME – the name of the user-defined type.
- MODEL – indicates if the attribute is to be sent to the analysis engine. A value of 0 indicates do not send; a value of 1 indicates send.
- VISIBLE – visibility flag. A value of 0 indicates invisible; a value of 1 indicates visible.
- ORDER_ID – not used.
- FORMAT – the output format for the vehicle attribute (for example, to put Page label in front of the number).

- TYPE_ENUM – the type of vehicle attribute. Values include:
  - 0 for integer
  - 1 for User Defined (specified by TYPE_EXTERNAL_NAME)
  - 2 for decimal
  - 3 for text
  - 4 for boolean
  - 5 for date
  - 6 for none

An Example

The following is an example of the data for a vehicle attribute record.

Table 3–30 Vehicle Attributes Example Data

Vehicle Attribute Name	Inactive	External Name	Description	Attribute Level	Vehicle External Name	Type External Name	Model	Visible	Order ID	Format	Type Enum
Page Location	0	page_location	Page Location	1	vehicle.circular	ude.page_location	1	0		{0}	0

Promote Interface Specifications

The following tables provide ordered lists of the contents of each of the Promote interface specifications. The specifications are organized into alphabetical order.

APE Price Elasticity Specification (BEE_APE_PRICE_ELASTICITY)

Table 3–31 APE Price Elasticity Standard Interface Specification[1]

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
DRIVER_APE_MERCH_NODE_EXT_ID	The external ID for the Driver Merchandise node.	String	200	Y
TARGET_APE_MERCH_NODE_EXT_ID	The external ID for the Target Merchandise node.	String	200	Y
LOC_LEVEL_DESC	The external ID for the external location level.	String	50	Y
LOC_CLIENT_LOAD_ID	The external ID for the location.	String	50	Y
ELASTICITY	The APE-calculated elasticity value.	Decimal	15,4	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

APE Promotion Elasticity Specification (BEE_APE_PROMO_ELASTICITY)

Table 3–32 APE Promotion Elasticity Standard Interface Specification[1]

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
DRIVER_APE_MERCH_NODE_EXT_ID	The external ID for the Driver Merchandise node.	String	200	Y
TARGET_APE_MERCH_NODE_EXT_ID	The external ID for the Target Merchandise node.	String	200	Y
LOC_LEVEL_DESC	The external ID for the external location level.	String	50	Y
LOC_CLIENT_LOAD_ID	The external ID for the location.	String	50	Y
PROMOTION_EXTERNAL_ATTR	A value generated by concatenating the source column name and its corresponding value.	String	200	Y
ELASTICITY	The APE-calculated elasticity value.	Decimal	15,4	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

Calendar Specification (ASH_CAL_TBL)

Table 3–33 Calendar Standard Interface Specification

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
EOP_CALENDAR_DT	Ending calendar date of the fiscal week (which is usually a Saturday).	Date in format YYYY-MM-DD	10	N
FISCAL_YR	Number of the fiscal year.	Integer	4	N
FISCAL_QTR	Number of fiscal quarter.	Integer	1	N
FISCAL_MO	Number of the fiscal month.	Integer	2	N
FISCAL_WK	Number of the fiscal week.	Integer	2	N
CALENDAR_WK	An alternative number for the calendar week (optional).	Integer	2	Y
SEASON	Season number associated with the week.	Integer	2	N

Demand Parameters Specification (ASH_PARAMETER_VALUES_TBL)

Table 3–34 Demand Parameters Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_LEVEL	The external merchandise level.	String	50	N
MERCHANDISE_KEY	In combination with the location key, identifies the item being marked down.	String	25	N
LOCATION_LEVEL	The external location level.	String	50	N
LOCATION_KEY	In combination with the merchandise key, identifies the item being marked down.	String	25	N
ITEM_ATTRIBUTE	The item attribute for the parameter (set to % by default).	String	100	N
PARAMETER_NAME	The name of the parameter. The names can be DEFAULT_ GAMMA, DEFAULT_ ALPHA, CRITICAL_ INVENTORY, or ZERO_INVENTORY.	String	50	N

Table 3–34 (Cont.) Demand Parameters Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
PARAMETER_VALUE	The value assigned to the parameter.	String	25	Y
AS_PARAMETER_ID	A number that uniquely identifies the record across all output tables and can be used to trace issues. It is not an analytical value.	Integer	32	Y
AS_VERSION_NUMBER	The version number for the current run of the output, which is set by APC and can be used to track versions.	String	20	Y

Future Price and Cost Specification (BEE_FUTURE_PRICE_COST)

Table 3–35 Future Price and Cost Standard Interface Specification

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MERCH_CLIENT_LOAD_ID	Customer's merchandise ID.	String	50	N
MERCH_LEVEL_DESC	Merchandise level description.	String	50	N
LOC_CLIENT_LOAD_ID	Customer's location ID.	String	50	N
LOC_LEVEL_DESC	Location level description.	String	50	N
EFFECTIVE_DT	The date of the change.	Date in format YYYY-MM-DD	10	N
PRICE	The changed price.	Decimal	15,4	N
COST	The changed cost.	Decimal	15,4	N

Images Specification (BEE_IMAGE)

Table 3–36 Images Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
NAME	Display name for image.	String	40	N
EXTERNAL_NAME	The ID for the image that is meaningful to the client. Unique across the images.	String	120	N
DESCRIPTION	An optional description of the image.	String	1000	Y
FILE_NAME	The filename of the image.	String	250	N
KEYWORDS	Keywords placeholder.	String	1000	Y

Table 3–36 (Cont.) Images Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
FILE_SIZE	The size of the image file.	Integer	10	Y
WIDTH	The image width.	Integer	10	Y
HEIGHT	The image height.	Integer	10	Y
RESOLUTION	The on-screen resolution of the image.	Integer	10	Y
DEPTH	The depth of the image.	Integer	10	Y
FILE_TYPE_ENUM	The image file type. Must be JPEG (0).	Integer	10	Y
MERCH_CLIENT_LOAD_ID	The client-specific category ID.	String	50	Y
LEVEL_DESC	The client-specific merchandise hierarchy level description.	String	50	Y

Inventory Specification (WK_HIST_SALES_INV_TBL)

Table 3–37 Inventory Standard Interface Specification [1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_KEY	The key from the merchandise hierarchy for the item.	String	25	N
LOCATION_KEY	The key from the location hierarchy for the item.	String	25	N
FISCAL_YR	The fiscal year of the sales record.	Integer	4	N
FISCAL_WK	The fiscal week of the sales record.	Integer	2	N
END_OH_QTY	The number of units of on-hand inventory at the end of the period.	Integer	12	N
END_OO_QTY	The number of inventory units in transit to the location at the end of the period.	Integer	12	Y
UNIT_RTL	The item's ticketed price at the end of the period.	Decimal	7,2	Y
UNIT_CST	The item's unit cost at the end of the period.	Decimal	7,2	Y
INIT_RTL	The item's ticketed price at the start of the season.	Decimal	7,2	Y

Table 3–37 (Cont.) Inventory Standard Interface Specification [1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
RECEIPT_QTY	The total store receipts (in units) from the distribution centers and from transfers.	Integer	12	Y
GRSS_SLS_QTY	The gross number of new units sold for the item at the location. This excludes returns.	Integer	12	Y
GRSS_SLS_AMT	The gross dollar amount of new sales for the item at the location during the period. This excludes returns.	Decimal	16,2	Y
NET_SLS_QTY	The net number of units sold of the item at the location. This includes returns.	Integer	12	Y
NET_SLS_AMT	The net dollar amount of sales for the item at the location during the period. This includes returns.	Decimal	16,2	Y
TOT_DSC_AMT	The total discount amount.	Decimal	16,2	Y
PROMO_MKDN_DSC_AMT	The total promotional markdown discount amount.	Decimal	16,2	Y
SELLIT_MKDN_DSC_AMT	The total sell-it discount amount.	Decimal	16,2	Y
CLR_DSC_AMT	The total clearance discount amount.	Decimal	16,2	Y
FREIGHT	The freight cost.	Decimal	16,2	Y
GRSS_PROFIT_AMT	The total gross margin (profit).	Decimal	16,2	Y
DUMMY	A dummy field.			
POS_SLS_QTY	The number of new units sold of the item at the location during the period.	Integer	12	Y
POS_SLS_AMT	The dollar amount of the new sales for the item at the location during the period.	Decimal	16,2	Y
MD_SALES_QTY	The units sold while on markdown.	Integer	12	Y

Table 3–37 (Cont.) Inventory Standard Interface Specification [1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MD_SALES_AMT	The sales dollars of the units sold while on markdown.	Decimal	16,2	Y
POS_MD_AMT	The total difference in weekly sales dollars between the promotional sales price and the inventory price.	Decimal	16,2	Y
PERM_MD_AMT	Includes distribution center, on hand, in transit, and store on hand.	Decimal	16,2	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

Items Specification (ASH_ITEMS_TBL)

Table 3–38 Items Standard Interface Specification[1]

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_KEY	Key for the item level in the merchandise hierarchy	String	25	N
LOCATION_KEY	Key for the item level in the location hierarchy	String	25	N
FIRST_RECEIPT_DATE	Receipt date is the date that an item first appears in a store or a distribution center (DC)	Date in format YYYY-MM-DD	10	Y
LAST_RECEIPT_DATE	Most recent date that an item was received in a store or a distribution center	Date in format YYYY-MM-DD	10	Y
VENDOR	Vendor that supplies merchandise to this location	String	25	Y
VENDOR_DESC	Vendor description	String	50	Y

Table 3–38 (Cont.) Items Standard Interface Specification[1]

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
UNIT_COST	Describes the merchandise's average unit cost (cost of inventory)	Decimal	22,2	N
SEASON_CODE	Retailer-specific season code, used to help determine seasonality	String	25	Y
FULL_PRICE	Original retail price of the merchandise	Decimal	22,2	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

Like Location Specification (BEE_PR_LIKE_LOCATION)

Table 3–39 Like Location Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
LOC_CLIENT_LOAD_ID	Customer's location ID.	String	50	N
LOC_LEVEL_DESC	Location level description.	String	50	N
LIKE_LOC_CLIENT_LOAD_ID	Customer's like location ID.	String	50	N
LIKE_LOC_LEVEL_DESC	Like location level description.	String	50	N

Like Merchandise Specification (BEE_PR_LIKE_MERCHANDISE)

Table 3–40 Like Merchandise Standard Interface Specification

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MERCH_CLIENT_LOAD_ID	Customer's merchandise ID.	String	50	N
MERCH_LEVEL_DESC	Merchandise level description.	String	50	N
LIKE_MERCH_CLIENT_LOAD_ID	Customer's like merchandise ID.	String	50	N
LIKE_MERCH_LEVEL_DESC	Like merchandise level description.	String	50	N
INACTIVE	Flag to identify whether or not the data is to be added or removed. 0 = Active and 1 = Inactive. Data to be removed must be set to 1.	Integer	1	N

Location Hierarchy Specification (ASH_LH_TBL)

Table 3–41 Location Hierarchy Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
HIERARCHY1_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY1_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY1_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY2_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY2_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY2_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY3_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY3_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY3_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY4_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY4_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY4_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY5_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY5_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY5_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY6_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY6_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY6_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY7_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY7_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY7_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY8_ID	ID for this level of the hierarchy.	String	25	Y

Table 3–41 (Cont.) Location Hierarchy Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
HIERARCHY8_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY8_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY9_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY9_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY9_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY10_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY10_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY10_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY11_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY11_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY11_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY12_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY12_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY12_DESC	Description of this level of the hierarchy.	String	50	Y

Location Hierarchy Attributes Specification (ASH_LH_ATTRS)

Table 3–42 Location Hierarchy Attributes Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
LOCATION_KEY	Unique identifier for location hierarchy.	String	25	N
LOCATION_LEVEL	Level within the location hierarchy.	String	50	N
MARKET_NAME	Market name.	String	50	Y
STORE_CITY	City.	String	50	Y
STORE_STATE	State.	String	50	Y
LOCATION_TYPE	Store class.	Integer	2	Y
STORE_NAME	Store name.	String	20	Y
STORE_POSTAL_CODE	Postal code.	String	20	Y

Table 3–42 (Cont.) Location Hierarchy Attributes Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
NSLS_SQFT	Net square footage.	Integer	6	Y
GRSS_SQFT	Gross square footage.	Integer	6	Y
OPEN_DT	Beginning of promotion.	Date in format YYYY-MM-DD	10	Y
CLOSE_DT	End of promotion.	Date in format YYYY-MM-DD	10	Y
CLIMATE	Climate code.	String	25	Y
STORE_FASHION_SEGMENT	Fashion segment code.	String	25	Y
STORE_AD_GROUP	Ad designation.	String	2	Y
STORE_SSC	Store service center (DC) number.	Integer	4	Y
STORE_CLSS_IND	Store class size.	String	25	Y
SSC_IND	Store service center indicator.	String	1	Y
STORE_CHST_1	Store characteristic 1.	String	20	Y
STORE_CHST_2	Store characteristic 2.	String	20	Y
STORE_CHST_3	Store characteristic 3.	String	20	Y
PRICING_GROUP	Pricing group.	String	20	Y
COMBO_STORE	Combo stores.	String	20	Y
TAXABILITY	Taxability.	String	20	Y
STORE_ZIP	Zip code.	Integer	9	Y
VOLUME_GR		String	50	Y
STORE_CLASS	Store class.	String	50	Y
GRS_ARE_SQFT	Gross square area.	Integer	9	NY

LH CDA Specification (ASH_LH_CDA_TBL)

Table 3–43 Location Hierarchy CDA Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
LOCATION_KEY	Unique identifier for location hierarchy.	String	25	N
LOCATION_LEVEL	Level within the location hierarchy.	String	50	N
ATTRIBUTE1		String	100	Y
ATTRIBUTE2		String	100	Y
ATTRIBUTE3		String	100	Y
ATTRIBUTE4		String	100	Y
ATTRIBUTE5		String	100	Y
ATTRIBUTE6		String	100	Y

Table 3–43 (Cont.) Location Hierarchy CDA Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
ATTRIBUTE7		String	100	Y
ATTRIBUTE8		String	100	Y
ATTRIBUTE1_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE2_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE3_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE4_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE5_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE6_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE7_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE8_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE1_NUMBER		Decimal	31,3	Y
ATTRIBUTE2_NUMBER		Decimal	31,3	Y
ATTRIBUTE3_NUMBER		Decimal	31,3	Y
ATTRIBUTE4_NUMBER		Decimal	31,3	Y
ATTRIBUTE5_NUMBER		Decimal	31,3	Y
ATTRIBUTE6_NUMBER		Decimal	31,3	Y
ATTRIBUTE7_NUMBER		Decimal	31,3	Y
ATTRIBUTE8_NUMBER		Decimal	31,3	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

LH Rename Specification (ASH_LHRENAME_TBL)

Table 3–44 Location Hierarchy Rename Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
OLD_LOCATION_KEY	Old unique identifier for location hierarchy.	String	25	N
NEW_LOCATION_KEY	New unique identifier for location hierarchy.	String	25	N
LOCATION_LEVEL	Level within the location hierarchy.	String	50	N

Merchandise Hierarchy Specification (ASH_MH_TBL)

Table 3–45 Merchandise Hierarchy Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
HIERARCHY1_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY1_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY1_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY2_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY2_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY2_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY3_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY3_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY3_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY4_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY4_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY4_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY5_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY5_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY5_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY6_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY6_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY6_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY7_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY7_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY7_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY8_ID	ID for this level of the hierarchy.	String	25	Y

Table 3–45 (Cont.) Merchandise Hierarchy Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
HIERARCHY8_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY8_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY9_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY9_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY9_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY10_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY10_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY10_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY11_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY11_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY11_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY12_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY12_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY12_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY13_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY13_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY13_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY14_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY14_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY14_DESC	Description of this level of the hierarchy.	String	50	Y
HIERARCHY15_ID	ID for this level of the hierarchy.	String	25	Y
HIERARCHY15_KEY	Key for this level of the hierarchy.	String	25	Y
HIERARCHY15_DESC	Description of this level of the hierarchy.	String	50	Y

Merchandise Hierarchy Attribute Specification (STAGE_MH_ATTRS_TBL)

Table 3–46 Merchandise Hierarchy Attributes Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_KEY	Unique identifier for merchandise hierarchy.	String	25	Y
MERCHANDISE_LEVEL	Level within the merchandise hierarchy.	String	50	Y
BRAND	ID of the brand.	String	50	Y
BRAND_DESC	Description of the brand.	String	50	Y
VENDOR	Number of the supplier. Contains the manufacturer number when the supplier is set as a warehouse.	String	50	Y
VENDOR_DESC	Description of the supplier.	String	50	Y
ITEM_SIZE	Physical size.	String	50	Y
CATEGORY	Category.	String	50	Y
CATEGORY_DESC	Category description.	String	50	Y
REPORT_CLIENT_ID	Client ID associated with report.	String	50	Y
START_DT	Beginning of plan.	Date in format YYYY-MM-DD	10	Y
FIRST_CREATE_DT	Date merchandise first introduced.	Date in format YYYY-MM-DD	10	Y
LAST_MODIFIED_DT	Time stamp of last modification.	Date in format YYYY-MM-DD	10	Y
PROD_LEVEL	Product level.	Integer	32	Y
COST	Wholesale cost.	Decimal	22,2	Y
RETAIL	Retail price.	Decimal	22,2	Y
PACK_SIZE	Pack size (inner).	Integer	22	Y
SIZE_RANGE_DESC	Description of size range.	String	50	Y
DISP_CODE	Disposition code.	String	2	Y
PURCH_TYPE	Basic (B); Fashion (F); Key (K).	String	1	Y
GRP_IN	Group indicator.	String	1	Y
PROD_TYPE	Product type.	String	30	Y
BRAND_NAME	Brand name.	String	50	Y
CNTL_RKL	Control RKL.	String	2	Y
COLL_ID	ID of collection.	Integer	6	Y
COLL_NAME	Name of collection.	String	30	Y

Table 3–46 (Cont.) Merchandise Hierarchy Attributes Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MSTR_COLL_IND	Master collection indicator.	String	1	Y
ORIG_IND	Origin indicator (Domestic/Import).	String	1	Y
WEIGHT	Weight.	Decimal	7,2	Y
COLOR_CNT	Number of colors per style.	Integer	2	Y
SIZE_GRP_DESC	Description of size group.	String	5	Y
LINE_PCT	Line percent.	Integer	3	Y
OOS_DATE	Season out-of-stock date.	Date in format YYYY-MM-DD	10	Y
VENDOR_STYLE	Vendor style number.	String	30	Y
ALLOC_FLAG	Allocate flag (RAP).	String	1	Y
FIRST_EFF_DT	The date on which the merchandise is first in effect.	Date in format YYYY-MM-DD	10	Y
LAST_EFF_DT	The date on which the merchandise is last in effect.	Date in format YYYY-MM-DD	10	Y
BRAND_TYPE	Not used.	String	1	Y
PROMO_EXCLUSION	Used to flag a record as excluded (Y) or not (N).	String	1	Y
MERCHANDISE_SUBTYPE	Season code.	String	20	Y
SIZE_RANGE_KEY	ID of size range.	String	25	Y
SIZE_KEY	ID of size.	String	25	Y
MERCHANDISE_FLOOR_SET	Subset of a season used to describe when an item is introduced to the floor.	String	20	Y
COLOR_FAMILY	Color family.	String	50	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

MH CDA Specification (ASH_MH_CDA_TBL)

Table 3–47 Merchandise Hierarchy CDA Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_KEY	Unique identifier for merchandise hierarchy.	String	25	N
MERCHANDISE_LEVEL	Level within the merchandise hierarchy.	String	50	N
ATTRIBUTE1		String	100	Y

Table 3–47 (Cont.) Merchandise Hierarchy CDA Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
ATTRIBUTE2		String	100	Y
ATTRIBUTE3		String	100	Y
ATTRIBUTE4		String	100	Y
ATTRIBUTE5		String	100	Y
ATTRIBUTE6		String	100	Y
ATTRIBUTE7		String	100	Y
ATTRIBUTE8		String	100	Y
ATTRIBUTE1_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE2_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE3_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE4_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE5_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE6_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE7_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE8_DATE		Date in format YYYY-MM-DD	10	Y
ATTRIBUTE1_NUMBER		Decimal	31,3	Y
ATTRIBUTE2_NUMBER		Decimal	31,3	Y
ATTRIBUTE3_NUMBER		Decimal	31,3	Y
ATTRIBUTE4_NUMBER		Decimal	31,3	Y
ATTRIBUTE5_NUMBER		Decimal	31,3	Y
ATTRIBUTE6_NUMBER		Decimal	31,3	Y
ATTRIBUTE7_NUMBER		Decimal	31,3	Y
ATTRIBUTE8_NUMBER		Decimal	31,3	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

MH Rename Specification (ASH_MHRENAME_TBL)

Table 3–48 Merchandise Hierarchy Rename Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
OLD_MERCHANDISE_KEY	Old unique identifier for merchandise hierarchy.	String	25	N
NEW_MERCHANDISE_KEY	New unique identifier for merchandise hierarchy.	String	25	N
MERCHANDISE_LEVEL	Level within the merchandise hierarchy.	String	50	N

Merchandise Thresholds Specification (BEE_MERCHANDISE_THRESHOLDS_TBL)

Table 3–49 Merchandise Thresholds Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MERCH_CLIENT_LOAD_ID	ID.	String	50	N
MERCH_LEVEL_DESC	Level equal to or higher than PROMOTE_MIN_LCD.	String	50	N
GREEN_THRESHOLD	Value for high confidence.	Integer	10	N
YELLOW_THRESHOLD	Value for medium confidence.	Integer	10	N
RED_THRESHOLD	Value for low confidence.	Integer	10	N

Offers Specification (BEE_OFFER)

Table 3–50 Offers Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
NAME	Display name for the offer.	String	40	N
INACTIVE	Activity flag. 0 = active. 1 = inactive.	Integer	1	N
EXTERNAL_NAME	The ID for the offer that is meaningful to the client. Unique across all offers.	String	120	N
DESCRIPTION	An optional description of the offer.	String	1000	Y
BUSINESS_RULE_CLASS_NAME	Instance of what class to use in validation.	String	250	Y
TYPE_EXTERNAL_NAME	Name of user defined type.	String	120	N
MODEL_CODE	Bit identifier for offer. Must be power of 2 (e.g., 0, 1, 2, 4, 8...).	Integer	10	N
FORMAT	Output format for offer (e.g., to put $ in front of number).	String	40	N
TYPE_ENUM	0 = integer; 1 = user-defined; 2 = decimal; 6 = none.	Integer	10	N

Period Attributes Specification (BEE_PERIODS_ATTR_TBL)

Table 3–51 Bee Periods Attributes Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
BEGIN_CALENDAR_DT	Beginning of the period.	Date in format YYYY-MM-DD	10	N
END_CALENDAR_DT	End of the period.	Date in format YYYY-MM-DD	10	N
DARKPERIOD_FLAG	Defines whether or not there are active promotions during specified dates. 1 = Yes; 0 = No.	String	1	N
DARKPERIOD_DESC	Optional description of dark period.	String	50	Y

Promotion Allocation Specification (BEE_PROMO_ALLOC)

Table 3–52 Promotion Allocation Standard Interface Specification [1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
PROMO_EXTERNAL_NAME	The ID for the promotion that is meaningful to the client.	String	120	N
MERCH_CLIENT_LOAD_ID	The client-specific category ID.	String	50	N
LEVEL_DESC	The client-specific merchandise hierarchy level description.	String	50	N
SPACE_ALLOCATION	The allocation for a given category.	Decimal	15,4	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

Promotion Campaign Specification (BEE_PROMO_CAMPAIGN)

Table 3–53 Promotion Campaign Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
NAME	A display name for the campaign.	String	40	N
DESCRIPTION	An optional description of the campaign.	String	1000	N
EXTERNAL_NAME	The ID for the campaign that is meaningful to the client. It is unique across all campaigns.	String	120	Y
BEGIN_DATE	The start date of the campaign.	Date in format YYYY-MM-DD	10	N
END_DATE	The end date of the campaign.	Date in format YYYY-MM-DD	10	N
INACTIVE	Activity flag. 0 = active. 1 = inactive.	Boolean (0,1)	1	N

Promotion Forecaster Specification (BEE_PROMO_FRCSTR)

Table 3–54 Merchandise Hierarchy Rename Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
PROMO_EXTERNAL_NAME	Promo ID.	String	120	N
OFFER_EXTERNAL_NAME	Offer ID.	String	120	N
ORDER_ID	Order ID.	Integer	10	N
EVENT_EXTERNAL_NAME	Event name.	String	120	N

Table 3–54 (Cont.) Merchandise Hierarchy Rename Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
FORECAST	1 = Forecast promo/offer. 0 = skip. Although this attribute is technically nullable, it is recommended that it not be left nullable.	Integer	1	N
FORCE	1 = Force the forecast promo/offer. 0 = skip. Although this attribute is technically nullable, it is recommended that it not be left nullable.	Integer	1	N
REFRESH	1 = refresh offer's criteria. 0 = skip. Although this attribute is technically nullable, it is recommended that it not be left nullable.	Integer	1	N

Promotion Offer Specification (BEE_PROMO_OFFER)

Table 3–55 Promotion Offer Standard Interface Specification [1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
INACTIVE	The status of the promotion offer. 1 = active. 0 = deleted. Default = 0.	Integer	1	Y
NAME	The display name for the offer.	String	40	N
EXTERNAL_NAME	The ID for the offer that is meaningful to the client.	String	120	N
DESCRIPTION	An optional description of the offer.	String	1000	Y
BEGIN_DATE	The start date for the offer.	Date in format YYYY-MM-DD	10	N
END_DATE	The end date for the offer.	Date in format YYYY-MM-DD	10	N
PROMO_EXTERNAL_NAME	The ID for the promotion that is meaningful to the client.	String	120	N
OFFER_EXTERNAL_NAME	The ID for the offer that is meaningful to the client.	String	120	N
EVENT_EXTERNAL_NAME	Associates an offer with other offers in the same event.	String	120	Y
UDV_EXTERNAL_NAME	The actual user-defined type value.	String	120	Y

Table 3–55 (Cont.) Promotion Offer Standard Interface Specification [1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
VALUE_INT	The integer value of the offer.	Integer	8	Y
VALUE_DEC	The currency value for the actual offer.	Decimal	15,4	Y
PAGE_NUM	The page of the offer.	Integer	8	Y
POS_NUM	The position of the offer.	Integer	4	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

Promotion Offer Attribute Specification (BEE_PROMO_OFFER_ATTR)

Table 3–56 Promotion Offer Attribute Standard Interface Specification [1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
PROMO_EXTERNAL_NAME	The ID for the promotion that is meaningful to the client.	String	120	N
PROMO_OFFER_EXTERNAL_NAME	The ID for the promotion offer that is meaningful to the client.	String	120	N
VEH_ATTR_EXTERNAL_NAME	The vehicle attribute name that is meaningful to the client.	String	120	N
UDV_EXTERNAL_NAME	The actual user-defined type value.	String	120	Y
VALUE_INT	The integer value of the offer.	Integer	8	Y
VALUE_DEC	The currency value for the actual offer.	Decimal	15,4	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

Promotion Offer Criteria Specification (BEE_PROMO_OFFER_CRITERIA)

Table 3–57 Promotion Offer Criteria Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
INACTIVE	Activity flag. 0 = active (default). 1 = deleted.	Integer	1	Y
EXTERNAL_NAME	ID of SKU list.	String	120	N
PROMO_EXTERNAL_NAME	ID of promotion for this criterion.	String	120	N

Table 3–57 (Cont.) Promotion Offer Criteria Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
PROMO_OFFER_EXTERNAL_NAME	ID fro promotion offer for this criterion.	String	120	N
CRITERION_TYPE	Offer criterion type. 0 = sku list. 1 = merchandise category. 2 = SKU.	Integer	4	N
SKU_LIST_EXTERNAL_NAME	If CRITERION_TYPE = 0, a meaningful ID.	String	120	Y
MERCH_CLIENT_LOAD_ID	If CRITERION_TYPE = 1, a meaningful ID.	String	50	Y
LEVEL_DESC	Level of the category.	String	50	Y
ATTRIBUTE_NAME	Restricts the criterion type. Values are RETAIL or VENDORNAME.	String	30	Y
ATTRIBUTE_VALUE	Restricts the category (for CRITERION_TYPE 1) by this attribute value.	String	50	Y
ATTRIBUTE_NAME2	Restricts the criterion type. Values are RETAIL or VENDORNAME.	String	30	Y
ATTRIBUTE_VALUE2	Restricts the category (for CRITERION_TYPE 1) by this attribute value.	String	50	Y
LOGICAL_OPERATOR	0 = or. 1 = and. Indicates how the two attribute values are combined.	Integer	4	Y
INCLUDE	1 = include specified SKU. 0 = exclude specified SKU.	Integer	1	N
LIST_TYPE	Buy List = 0; Get List = 1. Default is 0.	Integer	1	Y

Promotion Offer Merchandise Specification (BEE_PROMO_OFFER_MERCH)

Table 3–58 Promotion Offer Merchandise Standard Interface Specification [1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
PROMO_EXTERNAL_NAME	The ID for the promotion that is meaningful to the client.	String	120	N
PROMO_OFFER_EXTERNAL_NAME	The ID for the promotion offer that is meaningful to the client.	String	120	N
MERCH_CLIENT_LOAD_ID	The client-specific category ID.	String	50	N
LEVEL_DESC	The client-specific merchandise hierarchy level description.	String	50	N
FULL_PRICE	The price of the item.	Decimal	15,4	Y
PROMO_PRICE	The promotion price of the item.	Decimal	15,4	Y
COST	The actual cost of the item.	Decimal	15,4	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

Promotion Offer Store Specification (BEE_PROMO_STORE)

Table 3–59 Promotion Offer Store Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
PROMO_EXTERNAL_NAME	The ID for the promotion that is meaningful to the client.	String	120	N
LOC_CLIENT_LOAD_ID	The client-specific store hierarchy level description.	String	50	N
LEVEL_DESC	The client-specific hierarchy level description.	String	50	N

Promotions Specification (BEE_PROMOTIONS)

Table 3–60 Promotions Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
TYPE	Promotion type. 4 = historical promotion. 5 = pre-planned promotion.	Integer	4	N
INACTIVE	The status of the promotion. 0 = active. 1 = inactive. Default = 0.	Integer	1	Y
NAME	A display name for the promotion.	String	40	N
EXTERNAL_NAME	The ID for the promotion that is meaningful to the client. Unique across the promotion.	String	120	N
DESCRIPTION	An optional description of the promotion.	String	1000	Y
BEGIN_DATE	Start date of the promotion.	Date in format YYYY-MM-DD	10	N
END_DATE	End date of the promotion.	Date in format YYYY-MM-DD	10	N
TOTAL_COST	The total cost allocated to the promotion.	Decimal	15,4	Y
VEHICLE_EXTERNAL_NAME	The vehicle used when promoting items.	String	120	N

Table 3–60 (Cont.) Promotions Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
PAGES	The number of pages for the vehicle.	Integer	8	Y
EVENT_EXTERNAL_NAME	The name of the event used for the promotion.	String	120	N
CAMPAIGN_EXTERNAL_NAME	The name of the campaign used for the promotion.	String	120	N

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

Seasonal Trend (PR_PBL_TREND)

Table 3–61 Seasonal Trend Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
BEGIN_DALENDAR_DT	The beginning date for the data being analyzed.	Date in format YYYY-MM-DD	10	N
END_CALENDAR_DT	The end date for the data being analyzed.	Date in format YYYY-MM-DD	10	N
LOC_CLIENT_LOAD_ID	The external ID for the location.	String	50	N
LOC_LEVEL_DESC	The client-specific location level description.	String	120	N
USR_TREND	The value applied to the seasonal forecast.	Decimal	38,20	N

Seasonalities Specification (ASH_SEASONALITY_MAPS_TBL and ASH_SEASONALITY_VALUES_TBL)

The seasonalities interface populates two tables in Promote.

Table 3–62 Seasonalities (Maps) Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
PRIORITY	The search priority for the seasonality.	Integer		N
SEASONALITY_ID	The ID for the seasonality.	Integer		N
MERCHANDISE_LEVEL	Description of this level of the merchandise hierarchy.	String	50	N
MERCHANDISE_KEY	Key for this level of the merchandise hierarchy.	String	25	N
LOCATION_LEVEL	Description of this level of the location hierarchy.	String	50	N

Table 3–62 (Cont.) Seasonalities (Maps) Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
LOCATION_KEY	Key for this level of the location hierarchy.	String	25	N
ATTRIBUTE_VALUE_MASK	The search mask that specifies the season code and, optionally, the item attributes of the seasonality curves.	String	50	Y
AS_VERSION_NUMBER	The version number for the current run. Set by APC and used to track run versions.	String	20	Y

Table 3–63 Seasonalities (Values) Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
SEASONALITY_ID	The ID for the seasonality.	Integer		N
CALENDAR_DT	The date for the seasonality	Date in format YYYY-MM-DD	10	N
SEAS_INDX	The value of the seasonality for the date.	Decimal	11,4	Y
SEAS_ERR	For future use. Set to 0.	Decimal	11,4	Y
AS_PARAMETER_ID	A number that uniquely identifies the current record and that is used for tracking.	Integer		Y
AS_VERSION_NUMBER	The version number for the current run. Set by APC and used to track run versions.	String	20	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

SKU List Specification (BEE_SKU_LIST)

Table 3–64 SKU List Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
INACTIVE	The activity flag. 0 = active. 1 = deleted. Default = 0.	Integer	1	Y
NAME	The SKU list display name.	String	40	N
EXTERNAL_NAME	Meaningful ID for SKU list.	String	120	N
DESCRIPTION	Optional SKU list description.	String	1000	Y

SKU List Items Specification (BEE_SKU_LIST_ITEMS)

Table 3–65 SKU List Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
SKU_LIST_EXTERNAL_NAME	The parent SKU list ID.	String	120	N
MERCH_CLIENT_LOAD_ID	The customer's Like Merchandise ID.	String	50	N

Store Set Price Specification (BEE_STORE_SET_PRICE_TBL)

Table 3–66 Store Set Price Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MERCH_CLIENT_LOAD_ID	Client ID.	String	50	N
MERCH_LEVEL_DESC	Description of merchandise level.	String	50	N
STORE_SET_NAME	Name of store set.	String	50	N
STORE_SUBSET_NAME	Name of store subset.	String	50	N
PRICE	Price for version of promotion.	Decimal	15,4	N
COST	Associated cost.	Decimal	15,4	N

Store Sets Specification (BEE_STORE_SETS)

Table 3–67 Store Set Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
NEW_STORE_SET_NAME	New name for store set. Replaces old name.	String	50	N
OLD_STORE_SET_NAME	The existing name of the store set. Optional.	String	50	Y
STORE_SET_DESC	The description of the store set.	String	50	N
INACTIVE	Flag indicating activity status. 0 = inactive. 1 = active.	Integer	1	N
STORE_SET_TYPE	0	Integer	32	N
FIRST_EFF_DT	The date when the store set becomes active.	Date in format YYYY-MM-DD	10	N
LAST_MODIFIED_DATE	The date when the record is modified for the last time.	Date in format YYYY-MM-DD	10	Y
REMAIN_SUBSET_NAME	The name for the remaining subset for the associated store set.	String	50	Y

Store Subset Specification (BEE_STORE_SUBSETS)

Table 3–68 Store Subset Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
NEW_STORE_SUBSET_NAME	New name for the store subset. Replaces old name.	String	50	N
OLD_STORE_SUBSET_NAME	The existing name of the store subset.	String	50	Y
STORE_SUBSET_DESC	The description of the store subset.	String	50	N
STORE_SET_NAME	The name of the store set related to this store subset.	String	50	N
INACTIVE	Flag indicating activity status. 0 = inactive. 1 = active.	Integer	1	N
ORDER_SEQ	The position of the subset.	Integer	32	N

Store Subset Assignment Specification (BEE_STORE_SUBSET_ASSIGNMENT)

Table 3–69 Store Subset Assignment Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
LOC_CLIENT_LOAD_ID	The external ID for the location.	String	50	N
LEVEL_DESC	The external ID for the location level.	String	50	N
STORE_SUBSET_NAME	The name of the store subset for the location.	String	50	N
STORE_SET_NAME	The name of the store set for the location.	String	50	N

TAE Temp Metric Specification (BEE_TAE_TEMP_METRIC)

Table 3–70 TAE Temp Metric Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
RUN_ID	Execution ID.	Integer	32	Y
PROMO_ID	Internal promotion ID.	Integer	32	Y
AD_DATE	Promotion date.	Date in format YYYY-MM-DD	10	Y
PI_ID	Merchandise ID.	Integer	32	Y
LOCATION_ID	Location ID.	Integer	32	Y
AD_ITEM_PRICE	Metric.	Decimal	15,4	Y
AD_ITEM_ROSALE	Metric.	Integer	20	Y

Table 3–70 (Cont.) TAE Temp Metric Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
AD_ITEM_VISIT_RATE	Metric.	Integer	9	Y
AD_ITEM_SALES	Metric.	Decimal	15,4	Y
AD_ITEM_GM	Metric.	Decimal	15,4	Y
TTL_AD_DAYS	Metric.	Integer	9	Y
AD_ITEM_AC_SALES	Metric.	Decimal	15,4	Y
AD_ITEM_AC_GM	Metric.	Decimal	15,4	Y
AD_ITEM_PR_SALES	Metric.	Decimal	15.4	Y
AD_ITEM_PR_GM	Metric.	Decimal	15,4	Y
AD_NONAD_SALES	Metric.	Decimal	15,4	Y
AD_NONAD_GM	Metric.	Decimal	15,4	Y
BL_SUBST_CODE	Metric.	Integer	9	Y
BL_SUBST_ITEM	Metric.	Integer	32	Y
TTL_BASE_PERIODS	Metric.	Integer	9	Y
BL_ITEM_ROSALE	Metric.	Decimal	15,4	Y
BL_ITEM_SALES	Metric.	Decimal	15,4	Y
BL_ITEM_VISIT_RATE	Metric.	Decimal	15,4	Y
BL_ITEM_GM	Metric.	Decimal	15,4	Y
BL_ITEM_PRICE	Metric.	Decimal	15,4	Y
BL_ITEM_AC_SALES	Metric.	Decimal	15,4	Y
BL_ITEM_AC_GM	Metric.	Decimal	15,4	Y
BL_ITEM_PR_SALES	Metric.	Decimal	15,4	Y
BL_ITEM_PR_GM	Metric.	Decimal	15,4	Y
BL_NONAD_SALES	Metric.	Decimal	15,4	Y
BL_NONAD_GM	Metric.	Decimal	15,4	Y
AD_MB_ITEM_ONLY	Metric.	Integer	20	Y
AD_MB_ITEM_AD	Metric.	Integer	20	Y
AD_MB_ITEM_NONAD	Metric.	Integer	20	Y
AD_MB_ITEM_ADNONAD	Metric.	Integer	20	Y
AD_ITEM_OTHAD_ROS	Metric.	Integer	20	Y
STORE_BASE	Metric.	Decimal	15,4	Y

Table 3–70 (Cont.) TAE Temp Metric Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
AD_ITEM_NORMAL_PRICE	Metric.	Decimal	15,4	Y
AD_ITEM_AC_UNITS	Metric.	Decimal	15,4	Y
_ITEM_PR_UNITS	Metric.	Decimal	15,4	Y
AD_NONAD_UNITS	Metric.	Decimal	15,4	Y
BL_ITEM_AC_UNITS	Metric.	Decimal	15,4	Y
BL_ITEM_PR_UNITS	Metric.	Decimal	15,4	Y
BL_NONAD_UNITS	Metric.	Decimal	15,4	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

Transaction Log Specification (BEE_MB_DETAIL)

Table 3–71 Transaction Log Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
TXN_ID	Unique identifier for transaction.	String	50	N
TXN_DATE	Transaction date.	Date in format YYYY-MM-DD	10	N
LOC_CLIENT_LOAD_ID	ID for location where transaction occurred.	String	50	N
MERCH_CLIENT_LOAD_ID	ID of sold product.	String	50	N
UNIT_COST	Per-unit cost of sold product.	Decimal	15,4	Y
UNIT_NORMAL_PRICE	Per-unit non-promotional price of sold product.	Decimal	15,4	Y
UNITS_SOLD	The number of a given item that were purchased in the market basket.	Integer	9	Y
EXT_RETAIL_AMT	At-register price of product sold.	Decimal	15,4	Y
EXT_MARGIN_AMT	The amount that the price has been reduced if the item is on promotion for this type of item in the market basket.	Decimal	15,4	Y

Table 3–71 (Cont.) Transaction Log Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
AD_IND	Discount flag. 0 = none. 1 = on Ad (item was promoted). 2 = clearance.	Integer	9	Y
PROMO_TXN_CODE	Optional field. Offer code, coupon code, or other extended information.	String	100	Y

[1] For Decimal, the requirement is a number of a certain defined length and with a certain number of decimal places. For example, (22,2) is a number that can be up to 22 digits long and that can have two digits after the decimal point.

User Defined Type Specification (BEE_USER_DEFINED_TYPE)

Table 3–72 User Defined Type Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
TYPE_NAME	A display name for the type.	String	40	N
INACTIVE	Activity flag. 0 = active. 1 = inactive.	Integer	1	N
EXTERNAL_NAME	The ID for the type that is meaningful to the client. Unique across all types.	String	120	N
DESCRIPTION	An optional description of the offer.	String	1000	Y

User Defined Value Specification (BEE_USER_DEFINED_VALUE)

Table 3–73 User Defined Value Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
VALUE_NAME	A display name for the user-defined value.	String	40	N
INACTIVE	Activity flag. 0 = active. 1 = inactive.	Integer	1	N
EXTERNAL_NAME	The ID for the type that is meaningful to the client. Unique across all types.	String	120	N
TYPE_EXTERNAL_NAME	A string name of the user-defined type.	String	120	N
DESCRIPTION	Optional description of user-defined type.	String	1000	Y
ORDER_ID	Position of the element in an ordered list.	Integer	8	Y
EXTERNAL_CODE	The element's ID in the external system.	Integer	8	Y

Vehicle Specification (BEE_VEHICLE)

Table 3–74 Vehicle Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
VEHICLE_NAME	A display name for the vehicle.	String	40	N
INACTIVE	Activity flag. 0 = active. 1 = inactive.	Integer	1	N
EXTERNAL_NAME	The ID for the vehicle that is meaningful to the client. Unique across all vehicles.	String	120	N
DESCRIPTION	An optional description of the vehicle.	String	1000	Y
BUSINESS_RULE_CLASS_NAME	Instance of what class to use in validation.	String	250	Y
MODEL_CODE	Bit identifier for vehicle. Must be power of 2 (e.g., 0, 1, 2, 4, 8...).	Integer	10	N

Vehicle Attributes Specification (BEE_VEHICLE_ATTR)

Table 3–75 Vehicle Attributes Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
VEHICLE_ATTR_NAME	A display name for the vehicle attribute.	String	40	N
INACTIVE	Activity flag. 0 = active. 1 = inactive.	Integer	1	N
EXTERNAL_NAME	The ID for the vehicle attribute that is meaningful to the client. Unique across all vehicle attributes.	String	120	N
DESCRIPTION	An optional description of the vehicle attribute.	String	1000	Y
ATTRIBUTE_LEVEL	The level at which to show the attribute. 0 = vehicle. 1 = item.	Integer	1	Y

Table 3–75 (Cont.) Vehicle Attributes Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
VEHICLE_EXTERNAL_NAME	ID for the parent vehicle that is meaningful to the client. Unique across all vehicles.	String	120	N
TYPE_EXTERNAL_NAME	Name of user defined type.	String	120	N
MODEL	Flag indicating if attribute should be sent to analysis engine. 0 = not send. 1 = send.	Integer	1	N
VISIBLE	Visibility flag. 0 = invisible. 1 = visible.	Integer	1	Y
ORDER_ID	Not used.	Integer	8	Y
FORMAT	Output format for vehicle attribute (e.g., to put Page label in front of number).	String	40	N
TYPE_ENUM	The type of vehicle attribute: 0 = integer 1 = user defined 2= decimal 3 = text 4 = boolean 5 = date 6 = none	Integer	10	N

4

Standard Load

This chapter contains the following:

Introduction

This chapter describes the process to execute the standard load procedure, which transforms and loads retail data into the target database. It also includes standard load error messages and information about one-time data loads that are not part of the standard interface.

Standard Load Process

Promote provides two scripts that manage the process to stage, transform, and load data into the target database tables in the database. The data must be provided in flat files that meet the standard interface specifications. The variable length data in the files should be pipe-delimited. The files should be named to correspond to the names of the matching specification tables. For example, the calendar file should be named in a meaningful way (such as cal.txt) to correspond to ASH_CAL_TBL. No specific file extension is required for the input files.

The two scripts are located in %INSTALLATION_DIRECTORY%/modules/tools/bin. The first script, **pl_stage_file.sh**, stages the data from the flat files into the staging tables. The second script, **pl_load_data.sh**, loads the staged data into the Promote database. These two scripts are used if you need to customize the load dependency tree.

Each script contains options that can be customized. You can customize the options in the following ways (which are listed in order of precedence, with the command line having the highest precedence):

- Using the command line options
- Setting the customization values as environment variables in env.sh
- Setting the customization values in the user's environment

If you do not need to customize the load dependency tree, you can use the following two scripts:

- **pl_stage_client.sh** <*full_path_to_product_directory*> *DatasetFilename*
- **pl_load_client.sh** <*full_path_to_product_directory*>

The **pl_stage_client.sh** script calls **pl_stage_file.sh**. The **pl_load_client.sh** script calls **pl_load_data.sh**.

Environment Customization File

Here is an example of the environment customization file (**env.sh**):

```
#This is the environment customization file.
#Please define all customization values here.

#The mail client and address to send all messages to:
#MAIL=mailx
#REPORT_ADDRESS=error_mail@your_domain.com

#Number of parallel processes to run load procedures:
PARALLEL=2

#Directory with data control files:
#CONTROLDIR=/ASHschema/controlfiles

#Directory to store logs:
#LOGDIR=/tmp/load_logs

#Directory to move old logs to.
#If this variable is not set, the logs will be overwritten.
This folder is not required to exist and will be created at the time
#of archiving the logs.
#
#If all old logs should be preserved, it is possible to
#archive the files into a new unique folder, such as:
#LOGDIR_ARCHIVE=
#/tmp/load_logs/archived_logs_'date +%Y%m%d_%H%M_%S'
#
#If only the archive of the previous run is important, then
#archive the files into the same folder, such as:
#LOGDIR=/tmp/load_logs/archived_logs

#Number of errors to allow during load
ERROR_THRESHOLD=50
```

Staging Script: pl_stage_file.sh

Usage: pl_stage_file.sh *[OPTION]... [FILE]...*
Loads the files into the database.

Options:

Table 4–1 pl_stage_file.sh Options

-a DIR	--logdir_archive=DIR	directory to archive old log files
-c DIR	--controldir=DIR	directory with data control files
-e NUM	--errorthreshold=NUM	number of errors to allow in load (for DB2, it is a warning threshold)
-l DIR	--logdir=DIR	directory to store logs
-r DIR	--configroot=DIR	configuration root directory
-f FILENAME	--paramfile=FILENAME	filename of DB parameter file in configroot
-h	--help	displays help and exits

Load Script: pl_load_data.sh

Usage: pl_load_data.sh *[OPTION]... [LOADPROCEDURE]...*
Runs the load procedures in the database.

Options:

Table 4–2 pl_load_data.sh Options

-a DIR	--logdir_archive=DIR	directory to archive old log files
-e NUM	--errorthreshold=NUM	number of errors to allow in load (overwrites the procedure's default limit)
-l DIR	--logdir=DIR	directory to store logs
-r DIR	--configroot=DIR	configuration root directory
-f FILENAME	--paramfile=FILENAME	filename of DB parameter file in configroot
-rlo	--runloadonly	do not run pre and post step
-ne	--noexit	does not exit with a value
-h	--help	displays help and exits

Load Procedures

Here is a description of each load procedure, which includes the source table and the target table.

Load APE Price Elasticity

Procedure: com.profitlogic.db.beech.LoadApePriceElasticity

Source Table: BEE_APE_PRICE_ELASTICITY

Target Table: PR_APE_PRICE_ELASTICITY_TBL

Description: This procedure loads the price elasticity data generated by the APE component.

Load APE Promotion Elasticity

Procedure: com.profitlogic.db.beech.LoadApePromoElasticity

Source Table: BEE_APE_PROMO_ELASTICITY

Target Table: PR_APE_PROMO_ELASTICITY_TBL

Description: This procedure loads the promotion elasticity data generated by the APE component.

Load Calendars

Procedure: com.profitlogic.db.birch.LoadCalendars

Source Table: ASH_CAL_TBL

Target Table: PERIODS_TBL

Description: This procedure updates the PERIODS_TBL, which is seeded by Promote during installation. The following columns in PERIODS_TBL are updated:

- FISCAL_YR
- FISCAL_MO
- FISCAL_WK
- FISCAL_QUARTER
- FISCAL_HALF
- CALENDAR_YR
- CALENDAR_MO
- CALENDAR_WK
- CALENDAR_QUARTER
- SEASON (the rows derived from ASH_CAL_TBL)

Load Future Price and Cost

Procedure: com.profitlogic.db.beech.LoadFuturePriceCost

Source Table: BEE_FUTURE_PRICE_COST

Target Table: PR_FUTURE_PRICE_COST_TBL

Description: This procedure is responsible for loading future price and cost into the database.

Load Images

Procedure: com.profitlogic.db.beech.LoadImageMaster

Source Table: BEE_IMAGE

Target Table: PR_IMAGE

Description: This procedure is responsible for loading promotion offer images into the database for a sample image feed interface.

Load Inventory

Procedure: com.profitlogic.db.beech.load_weekly_history_data.load_at_all_levels

Source Table: WK_HIST_SALES_INV

Target Table: ACT_HIST_TBL_LVL_X

Description: This procedure is responsible for the inventory load.

Load Items

Procedure: com.profitlogic.db.birch.LoadItems

Source Tables:

- ASH_ITEMS_TBL
- ASH_ITEMS_CDA_TBL
- ASH_CP_TBL

Target Tables:

- ITEMS_TBL
- ITEMS_CDA_TBL

Description: This procedure inserts new data and updates existing data in ITEMS_TBL from ASH_ITEMS_TBL, based on the optimization level specified in ASH_CP_TBL (cross products information). It also inserts new data and updates existing data in ITEMS_CDA_TBL from ASH_ITEMS_CDA_TBL.

Load Like Location

Procedure: com.profitlogic.db.beech.LoadPromoLikeLocation

Source Table: BEE_PR_LIKE_LOCATION

Target Table: PR_LIKE_LOCATION

Description: This procedure is responsible for loading like stores.

Load Like Merchandise

Procedure: com.profitlogic.db.beech.LoadPromoLikeMerchandise

Source Table: BEE_PR_LIKE_MERCHANDISE

Target Table: PR_LIKE_MERCHANDISE

Description: This procedure is responsible for loading like items.

Load Location Hierarchy

Procedure: com.profitlogic.db.birch.LoadLocationHierarchy

Source Tables:

- ASH_LHL_TBL
- ASH_LH_TBL
- ASH_LH_CDA_TBL

Target Tables:

- LOCATION_HIERARCHY_TBL
- LOCATION_ATTR_TBL

Description: This procedure loads the entire location hierarchy, with the exception of the node (CHAIN) that is seeded by Promote during installation. It updates the location hierarchy based on the most recent information in ASH_LH_TBL and the levels specified in ASH_LHL_TBL. It completely re-loads LOCATION_ATTR_TBL with the most recent data from ASH_LH_CDA_TBL.

Load Location Hierarchy Attributes

Procedure: com.profitlogic.db.birch.LoadLHAttributes

Source Table: ASH_LH_ATTRS_TBL

Target Table: LOCATION_HIERARCHY_TBL

Description: This procedure is used to provide store details, such as opening and closing dates, in advance of the actual closing.

Load Location Table

Procedure: com.profitlogic.db.birch.LoadLHTbl

Source Table: LOCATION_HIERARCHY_TBL

Target Table: LOCATION_TBL

Description: This procedure completely re-loads LOCATION_TBL from LOCATION_HIERARCHY_TBL. LOCATION_TBL is a horizontally flattened view of the location hierarchy, used to improve the performance of other load procedures and Promote Planning.

Load LTClose Table

Procedure: com.profitlogic.db.birch.LoadLTCLOSE

Source Tables:

- LOCATION_TBL
- CLIENT_HIERARCHY_LEVELS_TBL

Target Table: LTCLOSE_TBL

Description: This procedure completely re-loads LTCLOSE_TBL from LOCATION_TBL using location hierarchy levels specified in CLIENT_HIERARCHY_LEVELS_TBL. TCLOSE_TBL is a vertically flattened view of the location hierarchy, containing each location node with all its parents. This table is used to improve the performance of other load procedures and Promote Planning.

Load LH Rename

Procedure: com.profitlogic.db.birch.LoadLHKeyRename

Source Table: ASH_LHRENAME_TBL

Target Table: LOCATION_HIERARCHY_TBL

Description: This procedure is responsible for moving locations within the location hierarchy. It updates CLIENT_LOAD_ID for a location node, based on the new LOCATION_KEY and LEVEL_DESC in ASH_LHRENAME_TBL.

Load Merchandise Hierarchy

Procedure: com.profitlogic.db.birch.LoadMerchandiseHierarchy

Source Tables:

- ASH_MHL_TBL
- ASH_MH_TBL
- ASH_MH_CDA_TBL

Target Tables:

- MERCHANDISE_HIERARCHY_TBL
- MERCH_ATTR_TBL
- PRODUCT_ITEMS_TBL

Description: This procedure loads the entire merchandise hierarchy, with the exception of the node (CHAIN) that is seeded by Promote during installation. It updates the merchandise hierarchy based on the most recent information in ASH_MH_TBL and the levels specified in ASH_MHL_TBL. It completely re-loads MERCH_ATTR_TBL with the most recent data from ASH_MH_CDA_TBL. It also updates PRODUCT_ITEMS_TBL according to the most recent merchandise hierarchy data.

Load Merchandise Hierarchy Attributes

Procedure: com.profitlogic.db.birch.LoadMerchandiseAttributes

Source Table: STAGE_MH_ATTRS_TBL

Target Table: MERCHANDISE_HIERARCHY_TBL

Load Merchandise Table

Procedure: com.profitlogic.db.birch.LoadMHTbl

Source Table: MERCHANDISE_HIERARCHY_TBL

Target Table: MERCHANDISE_TBL

Description: This procedure completely re-loads MERCHANDISE_TBL from MERCHANDISE_HIERARCHY_TBL. MERCHANDISE_TBL is a horizontally flattened view of the merchandise hierarchy, used to improve the performance of other load procedures and Promote Planning.

Load TClose Table

Procedure: com.profitlogic.db.birch.LoadTCLOSE

Source Tables:

- MERCHANDISE_TBL
- CLIENT_HIERARCHY_LEVELS_TBL

Target Table: TCLOSE_TBL

Description: This procedure completely re-loads TCLOSE_TBL from MERCHANDISE_TBL using merchandise hierarchy levels specified in CLIENT_HIERARCHY_LEVELS_TBL. TCLOSE_TBL is a vertically flattened view of the merchandise hierarchy, containing each merchandise node with all its parents. This table is used to improve the performance of other load procedures and Promote Planning.

Load MH Rename

Procedure: com.profitlogic.db.birch.LoadMHKeyRename

Source Table: ASH_MHRENAME_TBL

Target Table: MERCHANDISE_HIERARCHY_TBL

Description: This procedure is responsible for moving merchandise within the merchandise hierarchy. It updates CLIENT_LOAD_ID for a merchandise node, based on the new MERCHANDISE_KEY and LEVEL_DESC in ASH_MHRENAME_TBL.

Load Merchandise Thresholds

Procedure: com.profitlogic.db.beech.LoadMerchandiseThresholds

Source Table: BEE_MERCHANDISE_THRESHOLDS_TBL

Target Table: PR_MERCHANDISE_THRESHOLDS_TBL

Description: This interface defines thresholds for merchandise hierarchy levels.

Load Offers

Procedure: com.profitlogic.db.beech.LoadOfferMaster

Source Table: BEE_OFFER

Target Table: PR_OFFER

Description: This procedure is responsible for loading offers.

Load Parameters

Procedure: com.profitlogic.db.birch.LoadParameters

Source Table: ASH_PARAMETER_VALUES_TBL

Target Table: PARAMETER_VALUES_TBL

Description: This procedure truncates and reloads PARAMETER_VALUES_TBL. It populates client_search_levels_tbl with distinct parameter names and rank ordered sequence based on location and merchandise level sequences.

Load Period Attributes

Procedure: com.profitlogic.db.beech.LoadPromotePeriodsAttribute

Source Table: BEE_PERIODS_TBL

Target Table: PR_PERIOD_ATTR_TBL

Description: This procedure is used to define some Promote-specific attributes for periods.

Load Promotion Allocation

Procedure: com.profitlogic.db.beech.LoadPromoVehicleAlloc

Source Table: BEE_PROMO_ALLOC

Target Table:

- PR_PROMO_VEHICLE_ALLOC
- PR_PROMO_CAT

Description: This procedure is responsible for loading category allocations.

Load Promotion Campaign

Procedure: com.profitlogic.db.beech.LoadPromoCampaign

Source Table: BEE_PROMO_CAMPAIGN

Target Table: PR_CAL_ENT

Description: This procedure is responsible for loading campaigns.

Load PromotionForecaster

Procedure: com.profitlogic.db.beech.LoadPromoForecaster

Source Table: BEE_PROMO_FORECASTER

Target Table: PR_PROMO_FRCSTR_TBL

Description: This procedure is responsible for loading promotion forecast requests.

Load Promotion Offer

Procedure: com.profitlogic.db.beech.LoadPromoOffer

Source Table: BEE_PROMO_OFFER

Target Table: PR_PROMO_OFFER

Description: This procedure is responsible for loading promotion offers.

Load Promotion Offer Attributes

Procedure: com.profitlogic.db.beech.LoadPromoVehiclePagePositionOfferAttribute

Source Table: BEE_PROMO_OFFER_ATTR

Target Table: PR_PROMO_VEH_PG_POS_OFF_ATR

Description: This procedure is responsible for loading promotion offer attributes.

Load Promotion Offer Merchandise

Procedure: com.profitlogic.db.beech.LoadPromoOfferMerchandise

Source Table: BEE_PROMO_OFFER_MERCH

Target Table: PR_PROMO_OFFER_ITEM

Description: This procedure is responsible for loading offer items.

Load Promotion Offer Store

Procedure: com.profitlogic.db.beech.LoadPromoVehicleLocation

Source Table: BEE_PROMO_STORE

Target Table: PR_PROMO_VEH_LOC

Description: This procedure is responsible for loading promotion offer stores.

Load Promotions

Procedure: com.profitlogic.db.beech.LoadPromoMaster

Source Table: BEE_PROMOTIONS

Target Table: PR_PROMO

Description: This procedure is responsible for loading historical promotions.

Load Seasonalities

Procedure: com.profitlogic.db.birch.LoadSeasonalities

Source Tables:

- ASH_SEASONALITY_MAPS_TBL
- ASH_SEASONALITY_VALUES_TBL

Target Tables:

- SEASONALITY_MAPS_TBL
- SEASONALITY_VALUES_TBL

Description: This procedure truncates and reloads SEASONALITY_MAPS_TBL and SEASONALITY_VALUES_TBL. It populates client_search_levels_tbl with distinct parameter names and rank ordered sequence based on location and merchandise level sequences.

Load SKU List Cleanup Master

Procedure: com.profitlogic.db.beech.LoadSkuListCleanupMaster

Load SKU List Item Master

Procedure: com.profitlogic.db.beech.LoadSkuListItemMaster

Load SKU List Master

Procedure: com.profitlogic.db.beech.LoadSkuListMaster

Load Store Sets

Procedure: com.profitlogic.db.beech.LoadStoreSets

Source Tables:

- BEE_STORE_SETS

Target Tables:

- STORE_SETS_TBL

Description: This procedure is responsible for loading the store set definitions.

Load Store Set Prices

Procedure: com.profitlogic.db.beech.LoadStoreSetPrice

Source Tables:

- BEE_STORE_SET_PRICE

Target Tables:

- PR_STORE_SET_PRICE_TBL

Description: This procedure is responsible for loading merchandise prices for different pricing zones. The different pricing zones should be loaded via the Store Sets, Store Subsets, and Store Subset Assignment interfaces.

Load Store Subsets

Procedure: com.profitlogic.db.beech.LoadStoreSubsets

Source Tables:

- BEE_STORE_SUBSETS

Target Tables:

- STORE_SUBSETS_TBL

Description: This procedure is responsible for loading the subsets for a store set.

Load Store Subset Assignments

Procedure: com.profitlogic.db.beech.LoadStoreSubsetAssignments

Source Tables:

- BEE_STORE_SUBSET_ASSIGNMENT

Target Tables:

- STORE_SUBSET_ASSIGNMENT_TBL

Description: this procedure is responsible for loading the assignment of stores to a store set or store subset.

Load TAE Temp Metrics

Procedure: com.profitlogic.db.beech.LoadTAETemp.java

Source Table: BEE_TAE_TEMP_METRIC

Target Table: PR_TAE_TEMP_METRIC

Description: This procedure is responsible for loading TAE metrics into a temporary table.

Load Transaction Log

Procedure: com.profitlogic.db.beech.LoadMbDetail.java

Source Table: BEE_MB_DETAIL

Target Table: PR_MB_DETAIL

Description: This procedure is responsible for loading the Sales Transaction data.

Load UDE Types

Procedure: com.profitlogic.db.beech.LoadTypeMaster

Source Table: BEE_USER_DEFINED_TYPE

Target Table: PR_USER_DEFINED_TYPE

Description: This procedure is responsible for loading UDTs.

Load UDE Values

Procedure: com.profitlogic.db.beech.LoadValueMaster

Source Table: BEE_USER_DEFINED_VALUE

Target Table: PR_USER_DEFINED_VALUE

Description: This procedure is responsible for loading UDVs.

Load Vehicle

Procedure: com.profitlogic.db.beech.LoadVehicleMaster

Source Table: BEE_VEHICLE

Target Table: PR_VEHICLE

Description: This procedure is responsible for loading vehicles.

Load Vehicle Attributes

Procedure: com.profitlogic.db.beech.LoadVehicleAttributeMaster

Source Table: BEE_VEHICLE_ATTR

Target Table: PR_VEHICLE_ATTR

Description: This procedure is responsible for loading vehicle attributes.

Standard Load Error Handling

The Standard Load verifies the records in each staging table. Each record that fails the verification is removed from the staging table and placed in another table so that the load can continue and so that the failed records can be reviewed.

If a load procedure fails and the threshold is exceeded, you will see the message "The specified error threshold has been exceeded for this load procedure." If this occurs, you should correct the existing data problem and re-run the load procedure as well as any child load procedures (as shown in "Standard Load Procedures Order" on page 4-31).

The table containing the failed records is assigned a name that corresponds to the associated staging table. For example:

Table 4–3 Failed Records Table Names

Staging Table	Failed Record Table
ASH_CAL_TBL	ASH_CAL_TBL_BAD
BEECH_OFFER	BEECH_OFFER_BAD

The "BAD" table into which the failed records are inserted has the same structure as the corresponding staging table with the addition of the following four columns:

Table 4–4 Bad Table Columns

Column Name	Description	Data Type	Maximum Length	Nullable (Y/N)
ERROR_ROWID	The row ID that corresponds to the row ID in the staging table	Row ID		N
ERROR_CODE	The code for the verification	Integer		N
ERROR_DESC	Description of the error	String	1000	
ERROR_TIME	The time the error occurred	Timestamp		N

It is possible to place a threshold on the number of failed records in any staging table that will trigger a termination of the load. The default threshold values are hard-coded into Promote. In order to customize the threshold values, you must create a properties file and load it into Promote.

Error Handling Properties File

You can configure the threshold values for error handling in the properties file, **dbError.properties**. The values you set in this file override the corresponding Promote default values. The default value for the threshold of records failed is 100%. The default value for the total record threshold is 0%. Threshold values are expressed as a percentage. Note that the percentage symbol should not be included. Once you have created this file (which should be stored in **com/profitlogic/db/common/resources/dbError.properties** and called as a argument from there), you need to load it into the database schema using the procedure described on page 4-13.

Here is a sample **dbError.properties** file:

```
################################################################################################
#This properties file contains all error customizations
#
#Note:all thresholds should be satisfied in order for the load procedure to succeed
#
################################################################################################
#LoadPromotions error customizations
#
#Total error threshold is set to 0% of all records (default is 0%):
LoadPromotions.total.threshold=0
#
#Threshold of records failed with error 1205 should not exceed 100% (default is 100%):
LoadPromotions.1205.threshold=100
#
#Threshold of records failed with error 1207 should not exceed 100% (default is 100%):
LoadPromotions.1207.threshold=100
#########################################################################
```

In the **dbError.properties** file, you can set the total error threshold as well as a separate threshold for specific verifications. When configuring the error threshold for specific verifications, you use the error message number, as shown in Table 4–7, " Standard Load Error Messages" to indicate which verification you are setting the error threshold for. The sum of all the individual thresholds cannot exceed the total threshold.

Loading the dbError.properties File

Once you have created the **dbError.properties** file, you can load it, as follows:

dbpropertiesinstaller.sh <config_root> conf/com/profitlogic/db/common/resources/dbError.properties, where config_root is the root directory of the Promote configuration files.

The format for the file <db_connections_properties> is as follows:

For Oracle:

```
db.type=oracle
db.driver=oracle.jdbc.OracleDriver
db.url=jdbc:oracle:thin:@<db_host>:<db_port>:<db_SID>
db.password=<db_password>
```

where

<db_username> is	the username for the database connection
<db_password> is	the password for the database connection
<db_host> is	the host name of the database server

<db_port> is	the port number of the database server
<db_SID> is	the SID or SERVICE_NAME value for the database from the tnsnames.ora file.

Custom Errors

As part of the **dbError.properties** file, you can create custom verifications. Custom error codes have a reserved range of 50001 to 50100. You need to provide the text of the error message and a query that defines the verification. The pre-load verification (error messages 50000 and 50001 in the following sample) is run during the pre-load verification step. The post-load verification (error message 50002 in the following sample) is run during the post-load verification step. (For a list of the steps in the load procedure, see See "Standard Load Steps" on page 33.

Once you have modified the **dbError.properties** file to include custom verifications, you must load it into the database schema using the above command.

Here is a sample:

```
################################################################################
#Define custom PRE_LOAD verification errors with code 50000 and 50001
#(list of error codes separated by white spaces)
LoadPromotions.pre-load.custom-errors=50000 50001

#Error message:
LoadPromotions.pre-load.50000=Table ASH_CP_TBL is missing OPTIMIZATION levels
#Threshold (default is 100%):
#Note: the threshold affects only INSERT statements! If the statement is defined as a
#      SELECT, then the error will be triggered only if the query returns at least one row.
#      For any other type of statement amount of rows affected is not checked.
LoadPromotions.pre-load.50000.threshold=0
#INSERT statement should populate the "bad records" table with failed rows
#Note: in cases when the threshold is less than 100%, the INSERT statement should end
#       with a non-empty WHERE clause because the statement will be appended by an
#       additional condition.
LoadPromotions.pre-load.50000.query=                                    \
    SELECT 1 FROM %{YA_DUAL}%                                           \
     WHERE not exists (SELECT 1 FROM ash_cp_tbl                         \
                       WHERE intersect name = 'OPTIMIZATION')

#Error message:
LoadPromotions.pre-load.5000=No promotion is allowed after 01/01/2050
#Threshold (default is 100%):
#Note: the threshold affects only INSERT statements!
#      If the statement is defined as a SELECT, then the error will be
#      triggered only if the query returns at least one row.
#      For any other type of statement the number of rows is not checked.
LoadPromotions.pre-load.50001.threshold=0
#INSERT statement should populate the "bad records" table with failed rows
#Note: in cases when the threshold is less than 100%, the INSERT statement should end
#       with a non-empty WHERE clause because the statement will be appended by an
#      additional condition.
LoadPromotions.pre-load.50001.query=                                                    \
   INSERT INTO ash_promo_tbl_bad                                                        \
         (ERROR_ROWID, ERROR_CODE, ERROR DESC, ERROR_TIMESTAMP, merchandise_key,        \
          merchandise_level, location_key, location_level, promotion_key,              \
         promo_start_date, promo_end_date, promo_price, promo_perc_off,                 \
          promo_desc, promo_type, prono_excl_fg, promo_number, attribute1,              \
          attribute2, attribute3, attribute4, attribute5)                               \
```

```
   SELECT ROWID, 50001, 'Promo after 01/01/2050', %{YA_SYSDATE_AS_TIMESTAMP}%,           \
          merchandise_key,merchandise_level, location_key, location_level, promotion_key,\
         promo_start_date, promo_end_date, promo_price, promo_perc_off,                 \
          promo_desc, promo_type, prono_excl_fg, promo_number, attribute1,              \
          attribute2, attribute3, attribute4, attribute5)                               \
     FROM ash_promo_tbl                                                                 \
    WHERE promo_end_date >= %{YA_TODATE/''2050-01-01''/''YYYY-MM-DD''}%

################################################################################
# Define a custom POST_LOAD verification error with code 50002
# (list of error codes separated by spaces)
LoadPromotions.post-load.custom-errors=50002
LoadPromotions.post-load.50002=No promotion is allowed after 01/01/2050
#Note: If the statement is defined as a SELECT, then the error will be
#      triggered only if the query returns at least one row.
#      For any other type of statement the number of rows affected is not checked.
LoadPromotions.post-load.50002.query=                                                   \
    SELECT 1 FROM %{YA_DUAL}%                                                           \
    WHERE exists (SELECT 1 FROM planned_promos_tbl                                      \
                  WHERE end_dt >= %{YA_TO_DATE/''2050-01-01''/''YYYY-MM-DD''}%)
```

Error Handling Report

The standard load validates the data prior to loading the data into the target tables.

A customizable view, pl_load_status_vw, provides a report on the status of data validations. This view has the following default attributes:

Table 4–5 pl_load_status_vw Default Attributes

Attribute	Description
LOAD_PROCEDURE	The specific load procedure used
SOURCE	The staging table
DATA_VALIDATION_STATUS	Success - The number of failed records is less than the threshold set or Failure - The number or failed records exceeds the threshold set
NUM_BAD_RECORDS	The number of failed records in the failed record table

Here is a sample validation report:

Table 4–6 Sample Standard Load Data Validation Report

LOAD_PROCEDURE	SOURCE	DATA_VALIDATION_ STATUS	NUM_BAD_ RECORDS
LoadCHLevels	ASH_MHL_TBL	Success	0
LoadCHLevels	ASH_LHL_TBL	Success	0
LoadLocationHierarchyTbl	ASH_LH_TBL	Success	0
LoadOffer	BEE_OFFER	Success	0
LoadCalendars	ASH_CAL_TBL	Success	0
LoadVehicle	BEE_VEHICLE	Success	0

Table 4–6 (Cont.) Sample Standard Load Data Validation Report

LOAD_PROCEDURE	SOURCE	DATA_VALIDATION_ STATUS	NUM_BAD_ RECORDS
LoadLocationHierarchy	ASH_LH_CDA_TBL	Success	0
LoadMerchandiseHierarchy	ASH_MH_CDA_TBL	Success	0
No transformation of data	ASH_CP_TBL	Success	0
LoadLHKeyRename	ASH_LHRENAME_TBL	Success	0
LoadMHKeyRename	ASH_MHRENAME_TBL	Success	0
LoadVehicleAttr	BEE_VEHICLE_ATTR	Success	0
LoadUDType	BEE_USER_DEFINED_TYPE	Success	0
LoadUDValue	BEE_USER_DEFINED_VALUE	Failure	50
Load Promotions	BEE_PROMOTIONS	Success	0

To generate an output file that can be emailed to interested users or integrated into production scripts, use the following script. The script writes to the standard output, which can be redirected to a file. Note that the optional WHERE clause, including the WHERE keyword itself, should be enclosed in quotes.

bash pl_load_status.sh *-r <configroot> -w <whereclause>*

where

-r DIR	--configroot=DIR	The configuration root directory
-w WHERE	--whereclause=WHERE	An optional clause used to filter specific information in the report
-h	--help	Displays help and exits

Standard Load Error Messages

The following are the error messages that may be generated during the standard load procedure.

Table 4–7 Standard Load Error Messages

Number	Error Message
System Errors	
0	The program has completed successfully.
10	An unspecified error has occurred.
20	An SQL exception has occurred.
30	A Java exception has occurred.
40	The exception limit has been exceeded.
50	The specified error threshold has been exceeded in this load procedure.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
Common Errors	
100	At least one node in the hierarchy has more than one parent.
101	The number of levels in the levels table does not match the data from the source table.
102	The CHAIN level does not exist in the target table.
104	The levels table is empty.
105	The sequence for the CHAIN level should be defined as 1 in the levels table.
106	At least one node in the hierarchy has more than one hierarchy ID or description.
Load CH Levels Errors	
200	The cross-products information table (ASH_CP_TBL) does not have all the required records.
201	In the cross-products information table (ASH_CP_TBL), at least one INTERSECT_NAME has a value of NULL. An INTERSECT_NAME cannot have a value of NULL.
202	A duplicate INTERSECT_NAME has been found in the cross-products information table (ASH_CP_TBL).
203	Invalid INTERSECT_NAME has been found in the cross-products information table (ASH_CP_TBL) or not all necessary values (OPTIMIZATION, WORKSHEET, SALES, or CLUSTER) have been supplied.
204	The cross-products information table (ASH_CP_TBL) is empty.
205	In the cross-products information table (ASH_CP_TBL), at least one merchandise level has a value of NULL. A merchandise level cannot have a value of NULL.
206	In the cross-products information table (ASH_CP_TBL), at least one location level has a value of NULL. A location level cannot have a value of NULL.
Load Calendars Errors	
1000	In the calendar table (ASH_CAL_TBL), at least one fiscal year does not have between 52 and 53 weeks.
1001	In the calendar table (ASH_CAL_TBL), at least one fiscal year does not include twelve fiscal months.
1002	In the calendar table (ASH_CAL_TBL), at least one fiscal week has an End of Period (EOP) that is not Saturday.
1003	In the calendar table (ASH_CAL_TBL), at least one fiscal month is not in the range 1 - 12.
1004	In the calendar table (ASH_CAL_TBL), at least one fiscal week is not in the range 1 -53.
1005	In the calendar table (ASH_CAL_TBL), at least one fiscal year has a value of NULL. A fiscal year cannot have a value of NULL.
1006	In the calendar table (ASH_CAL_TBL), at least one fiscal month has a value of NULL. A fiscal month cannot have a value of NULL.
1007	In the calendar table (ASH_CAL_TBL), at least one fiscal week has a value of NULL. A fiscal week cannot have a value of NULL.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
1008	In the calendar table (ASH_CAL_TBL), at least one fiscal season has a value of NULL. A fiscal season cannot have a value of NULL.
1009	In the calendar table (ASH_CAL_TBL), at least one End of Period (EOP) has a value of NULL. A End of Period (EOP) cannot have a value of NULL.
1010	In the calendar table (ASH_CAL_TBL), at least one fiscal quarter has a value of NULL. A fiscal quarter cannot have a value of NULL.
1011	In the calendar table (ASH_CAL_TBL), at least one week end day does not match the existing week end date.
1012	The week end day is null.
1013	The week begin day is null.
Load Location Hierarchy Errors	
1500	In the location hierarchy CDA staging table (ASH_LH_CDA_TBL), at least one location key has a value of NULL. A location key cannot have a value of NULL.
1501	In the location hierarchy CDA staging table (ASH_LH_CDA_TBL), at least one location level has a value of NULL. A location level cannot have a value of NULL.
1502	In the location hierarchy levels table (ASH_LHL_TBL), at least one location level has a value of NULL. A location level cannot have a value of NULL.
1503	In the location hierarchy levels table (ASH_LHL_TBL), at least one level sequence level has a value of NULL. A level sequence cannot have a value of NULL.
1504	In the location hierarchy levels table (ASH_LHL_TBL) the entries in LEVEL_SQC are not sequential.
1505	The location hierarchy levels table (ASH_LHL_TBL) should have sequence starting with 1.
1506	In the location hierarchy levels table (ASH_LHL_TBL), CHAIN is not assigned a sequence value (LEVEL_SQC) of 1.
1507	In the merchandise hierarchy table (ASH_MH_TBL), null values were detected in the hierarchy stage key columns.
Load Location Hierarchy Key Rename Errors	
1600	In the location hierarchy rename table (ASH_LHRENAME_TBL), at least one old location key has a value of NULL. A location key cannot have a value of NULL.
1601	In the location hierarchy rename table (ASH_LHRENAME_TBL), at least one new location key has a value of NULL. A location key cannot have a value of NULL.
1602	In the location hierarchy rename table (ASH_LHRENAME_TBL), at least one location level has a value of NULL. A location level cannot have a value of NULL.
1603	The old location key in the location hierarchy rename table (ASH_ LHRENAME_TBL) contains duplicate values.
1604	The new location key in the location hierarchy rename table (ASH_ LHRENAME_TBL) contains duplicate values.
1605	The new location key in the location hierarchy rename table (ASH_ LHRENAME_TBL) is already present in the location hierarchy.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
Load Merchandise Hierarchy Errors	
2001	NOT NULL has already been set for the merchandise hierarchy table (ASH_MH_TBL) stage key columns.
2002	In the merchandise hierarchy table (ASH_MH_TBL), an error dropping the unique index occurred.
2501	In the merchandise hierarchy table (ASH_MH_TBL), null values were detected in the hierarchy stage key columns.
2502	The merchandise hierarchy levels table (ASH_MHL_TBL) is empty.
2503	In the merchandise hierarchy levels table (ASH_MHL_TBL) the entries in LEVEL_SQC are not sequential.
2504	The merchandise hierarchy levels table (ASH_MHL_TBL) should contain a sequence starting with 1.
2505	In the merchandise hierarchy levels table (ASH_MHL_TBL), CHAIN is not assigned a sequence value (LEVEL_SQC) of 1.
2506	The merchandise hierarchy staging table contains duplicate values at the lowest key level.
2507	The merchandise hierarchy table (ASH_MH_TBL) contains a child node with more than one parent node.
2508	The merchandise hierarchy cda staging table (ASH_MH_CDA_TBL) contains at least one combination of MERCHANDISE_KEY and MERCHANDISE_LEVEL that is not unique.
2509	In the merchandise hierarchy CDA staging table (ASH_MH_CDA_TBL), at least one merchandise key has a value of NULL. A merchandise key cannot have a value of NULL.
2510	The merchandise hierarchy rename table (ASH_MHRENAME_TBL) contains duplicate values for OLD_MERCHANDISE_KEY.
2511	In the merchandise hierarchy levels table (ASH_MHL_TBL), at least one merchandise level has a value of NULL. A merchandise level cannot have a value of NULL.
2512	In the merchandise hierarchy levels table (ASH_MHL_TBL), at least one level sequence level has a value of NULL. a level sequence cannot have a value of NULL.
Load MH Key Rename Errors	
2600	In the merchandise hierarchy rename table (ASH_MHRENAME_TBL), at least one old merchandise key has a value of NULL. A merchandise key cannot have a value of NULL.
2601	In the merchandise hierarchy rename table (ASH_MHRENAME_TBL), at least one new merchandise key has a value of NULL. A merchandise key cannot have a value of NULL.
2602	In the merchandise hierarchy rename table (ASH_MHRENAME_TBL), at least one merchandise level has a value of NULL. A merchandise level cannot have a value of NULL.
2603	The old merchandise key in the merchandise hierarchy rename table (ASH_MHRENAME_TBL) contains duplicate values.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
2604	The new merchandise key in the merchandise hierarchy rename table (ASH_MHRENAME_TBL) contains duplicate values.
2605	The new merchandise key in the merchandise hierarchy rename table (ASH_MHRENAME_TBL) is already present in the merchandise hierarchy.
Load MHTbl Errors	
6101	The MERCHANDISE_HIERARCHY_TBL table has no CHAIN record (where PARENT_MERCHANDISE_ID is NULL).
6102	The MERCHANDISE_HIERARCHY_TBL table has more than one record with PARENT_MERCHANDISE_ID = NULL (multiple CHAIN records).
Promote LoadTAE Errors	
8001	TAE output has invalid Merchandise CLIENT_LOAD_ID. It should match what is defined in MERCHANDISE_HIERARCHY_TBL.
8002	TAE output has invalid Location CLIENT_LOAD_ID. It should match what is defined in LOCATION_HIERARCHY_TBL.
8003	TAE output has an invalid Promotion. It should match what is defined in PR_PROMOTIONS
8004	TEMP TAE output has an invalid Merchandise CLIENT_LOAD_ID. It should match what is defined in MERCHANDISE_HIERARCHY_TBL.
8005	TEMP TAE output has an invalid Location CLIENT_LOAD_ID. It should match what is defined in LOCATION_HIERARCHY_TBL.
8006	TEMP TAE output has an invalid Promotion. It should match what is defined in PR_PROMOTIONS.
Load Parameters Errors	
8101	The merchandise key in ASH_PARAMETER_VALUES_TBL cannot be null.
8102	The merchandise level in ASH_PARAMETER_VALUES_TBL cannot be null.
8103	The location key in ASH_PARAMETER_VALUES_TBL cannot be null.
8104	The location level in ASH_PARAMETER_VALUES_TBL cannot be null.
8105	The item attribute in ASH_PARAMETER_VALUES_TBL cannot be null.
8106	The parameter name in ASH_PARAMETER_VALUES_TBL cannot be null.
8107	Merchandise found in ASH_PARAMETER_VALUES_TBL that does not exist in MERCHANDISE_HIERARCHY_TBL.
8108	Location found in ASH_PARAMETER_VALUES_TBL that does not exist in LOCATION_HIERARCHY_TBL.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
Load Seasonalities Errors	
8301	The merchandise key in ASH_SEASONALITY_MAPS_TBL cannot be null.
8302	The merchandise level in ASH_SEASONALITY_MAPS_TBL cannot be null.
8303	The location key in ASH_SEASONALITY_MAPS_TBL cannot be null.
8304	The location level in ASH_SEASONALITY_MAPS_TBL cannot be null.
8305	Merchandise found in ASH_SEASONALITY_MAPS_TBL that does not exist in MERCHANDISE_HIERARCHY_TBL.
8306	Location found in ASH_SEASONALITY_MAPS_TBL that does not exist in LOCATION_HIERARCHY_TBL.
8307	A NULL priority has been found.
8308	A NULL seasonality ID in maps has been found.
8309	A NULL seasonality ID in values has been found.
8310	A NULL calendar date has been found.
8311	A NULL AS_VERSION has been found in the seasonality maps.
8312	More than one distinct AS_VERSION value has been found in seasonality maps.
8313	A NULL AS_VERSION has been found in seasonality values.
8314	More than one distinct AS_VERSION value has been found in seasonality values.
Promote LoadPromoMaster Errors	
8401	The Date defined for the promotion is NOT defined in the Fiscal Calendar.
Promote LoadTypeMaster Errors	
8501	The NAME of Attribute Type should NOT be NULL.
8502	The INACTIVE status of Attribute Type should NOT be NULL. It should be 0 or 1.
8503	The EXTERNAL NAME of Attribute Type should NOT be NULL.
8504	The EXTERNAL NAME of Attribute Type should be unique within a single load file.
Promote LoadValueMaster Errors	
8601	The NAME of Attribute Value should NOT be NULL.
8602	The INACTIVE status of Attribute Value should NOT be NULL. It should be 0 or 1.
8603	The EXTERNAL NAME of Attribute Value should NOT be NULL.
8604	The EXTERNAL NAME of Attribute Value should be unique within a single load file.
8605	The EXTERNAL CODE of Attribute Value should NOT be NULL.
8606	The Attribute Value TYPE is invalid.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
Promote LoadOfferMaster Errors	
8701	The NAME of the Offer should NOT be NULL.
8702	The INACTIVE status of Offer should NOT be NULL. It should be 0 or 1.
8703	The EXTERNAL NAME of Offer should NOT be NULL.
8704	The Type for Offer, if specified, should be a valid User Defined Type.
8705	The MODEL_TYPE of Offer should NOT be NULL.
Promote LoadVehicleMaster Errors	
8801	The NAME of Vehicle should NOT be NULL.
8802	The INACTIVE status of Vehicle should NOT be NULL. It should be 0 or 1.
8803	The EXTERNAL NAME of Vehicle should NOT be NULL.
8804	The MODEL_TYPE of Vehicle should NOT be NULL.
Promote LoadVehicleAttributeMaster Errors	
8901	The NAME of Vehicle Attribute should NOT be NULL.
8902	The INACTIVE status of Vehicle Attribute should NOT be NULL. It should be 0 or 1.
8903	The EXTERNAL NAME of Vehicle Attribute should NOT be NULL.
8904	The Type for Vehicle Attribute, if specified, should be a valid User Defined Type.
8905	The Vehicle used in the Vehicle Attribute is invalid.
8906	The Attribute level is invalid. It should be 0 - promotions or 1 - item.
8907	The value for MODEL is invalid. It should be 1 - True or 0 - False.
8908	The value for VISIBLE is invalid. It should be 1 - True or 0 - False.
Promote LoadPromoteVehicle Errors	
9001	The NAME of Promotion Vehicle should NOT be NULL.
9002	The INACTIVE status of Promotion Vehicle Attribute should NOT be NULL. It should be 0 or 1.
9003	The BEGIN DATE of Promotion Vehicle should NOT be NULL.
9004	The END DATE of Promotion Vehicle should NOT be NULL.
9005	The PROMOTION specified in Promotion Vehicle is invalid.
9006	VEHICLE specified in Promotion Vehicle is invalid
9007	The promotion a promotion vehicle is assigned to cannot be changed.
Promote LoadPromoteVehicleAttr Errors	
9101	The INACTIVE status of Promotion Vehicle Attribute should NOT be NULL. It should be 0 or 1.
9102	The PROMOTION specified in Promotion Vehicle Attribute is invalid.
9103	The VEHICLE specified in Promotion Vehicle Attribute is invalid.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
9104	The VEHICLE ATTRIBUTE specified in Promotion Vehicle is invalid.
9105	Either the UDV_EXTERNAL_NAME or the VALUE_INT should be specified.
9106	The UDV_EXTERNAL_NAME used is not a valid User Defined Value.
9107	The Vehicle used is not defined for this Promotion.
9108	The Vehicle Attribute used is not defined for this Vehicle.
9109	Type of UDV and Vehicle Attribute do not match.
Promote LoadPromoteItem Errors	
9201	The INACTIVE status of the Promotion Item should NOT be NULL. It should be 0 or 1.
9202	The BEGIN DATE of the Promotion Vehicle should NOT be NULL.
9203	The END DATE of the Promotion Vehicle should NOT be NULL.
9204	The PROMOTION specified in Promotion Item is invalid.
9205	The OFFER specified in Promotion Item is invalid.
9206	The MERCH_CLIENT_LOAD_ID specified in Promotion Item is invalid.
9207	Either the UDV_EXTERNAL_NAME or VALUE_INT should be specified
9208	The UDV_EXTERNAL_NAME used is not a valid User Defined Value.
9209	Type of UDV and Offer do not match.
Promote LoadPromoteItemVehicle Errors	
9301	The INACTIVE status of Promotion Item Vehicle should NOT be NULL. It should be 0 or 1.
9302	The PROMOTION specified in Promotion Item Vehicle is invalid.
9303	The MERCH_CLIENT_LOAD_ID specified in Promotion Item Vehicle is invalid.
9304	The Promotion Item specified in Promotion Item Vehicle is invalid.
9305	The Vehicle specified in Promotion Item Vehicle is invalid.
9306	The Promotion Vehicle specified in Promotion Item Vehicle is invalid.
Promote LoadPromoteVehicleAttr Errors	
9401	The INACTIVE status of Promotion Item Vehicle Attribute should NOT be NULL. It should be 0 or 1.
9402	The PROMOTION specified in Promotion Item Vehicle Attribute is invalid.
9403	The VEHICLE specified in Promotion Item Vehicle Attribute is invalid.
9404	The VEHICLE ATTRIBUTE specified in Promotion Item Vehicle is invalid.
9405	Either the UDV_EXTERNAL_NAME or VALUE_INT should be specified.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
9406	The UDV_EXTERNAL_NAME used is not a valid User Defined Value.
9407	The Vehicle used is not defined for this Promotion.
9408	The Vehicle Attribute used is not defined for this Vehicle.
9409	Type of UDV and Vehicle Attribute do not match.
Promote Interface Load Parse Errors	
9501	Invalid file format for Promote Interface.
9502	Invalid file format for Promote Item Interface.
Promote Vehicle Allocation Errors	
9601	The Promotion specified in Promo Vehicle Allocation File cannot be NULL.
9602	The Promotion specified in Promo Vehicle Allocation File is invalid.
9603	The Vehicle specified in Promo Vehicle Allocation File is invalid.
9604	The Promotion Vehicle specified in Promo Vehicle Allocation File is invalid.
9605	The Merchandise specified in Promo Vehicle Allocation File is invalid.
9606	The Merchandise specified in Promo Vehicle Allocation File cannot be NULL.
Promotions Errors *(from the BEECH schema)*	
12001	The NAME of Promotions should NOT be NULL.
12002	The EXTERNAL_NAME of Promotions should NOT be NULL.
12003	The BEGIN_DATE of Promotions should NOT be NULL.
12004	The END_DATE of Promotions should NOT be NULL.
12005	TYPE must be HISTORIC.
12006	TYPE must be PREPLANNED.
12007	TYPE of Promotions should NOT be NULL.
12008	The value for INACTIVE should NOT be NULL.
12009	Parent record is invalid.
12010	Child record is invalid.
12011	The invalid promotion cannot be updated. The promotion must be active.
Location Attributes Load Errors *(from the BASE schema)*	
12001	The LOCATION_LEVEL does not exist in CLIENT_HIERARCHY_LEVELS_TBL.
12002	The LOCATION_KEY does not exist in LOCATION_HIERARCHY_TBL.
12003	The LOCATION_LEVEL cannot be NULL.
12004	The LOCATION_KEY cannot be NULL.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
Promo Store Errors	
12101	The PROMO_EXTERNAL_NAME should NOT be NULL.
12102	The PROMO_VEHICLE_EXTERNAL_NAME should NOT be NULL.
12103	The LOC_CLIENT_LOAD_ID should NOT be NULL.
12104	The PROMO_EXTERNAL_NAME is not valid.
12105	The LOC_CLIENT_LOAD_ID is not valid.
Promo Offer Errors	
12201	The NAME should NOT be NULL.
12202	The EXTERNAL_NAME should NOT be NULL.
12203	The BEGIN_DATE should NOT be NULL.
12204	The END_DATE should NOT be NULL.
12205	The PROMO_EXTERNAL_NAME should NOT be NULL.
12206	The PROMO_EXTERNAL_NAME is not valid. The correct promotion should be included in the current data feed.
12207	The OFFER_EXTERNAL_NAME should NOT be NULL.
12208	The OFFER_EXTERNAL_NAME is not valid.
12209	The UDVEXTERNAL_NAME is not valid.
12210	The value for INACTIVE should NOT be NULL.
12211	The value for PAGE_NUM or POS_NUM is not valid. Either both or none must be specified.
12212	The value for POS_NUM is not valid.
12213	The promo offer record is invalid. The promo ID and the external name must be unique.
12214	The related offer value (VALUE_INT, VALUE_DEV, or UDV_ EXTERNAL_NAME) should not be NULL. See the TYPE_ENUM field.
12215	The unrelated offer values (VALUE_INT, VALUE_DEV, or UDV_ EXTERNAL_NAME) should be NULL. See the TYPE_ENUM field.
12216	The begin date and the end date should match the dates for the corresponding promotion.
Promo Offer Merchandise Errors	
12301	The PROMO_EXTERNAL_NAME should NOT be NULL.
12302	The OFFER_EXTERNAL_NAME should NOT be NULL.
12303	The MERCH_CLIENT_LOAD_ID and LEVEL_DESC should not be null.
12304	The MERCH_CLIENT_LOAD_ID and LEVEL_DESC combination is not valid.
Promo Campaign Errors	
12401	The NAME should not be NULL.
12402	The EXTERNAL_NAME should not be NULL.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
12403	The BEGIN_DATE should NOT be NULL.
12404	The END_DATE should NOT be NULL.
Promo Offer Attribute Errors	
12501	The PROMO_EXTERNAL_NAME should NOT be NULL.
12502	The PROMO_OFFER_EXTERNAL_NAME should not be NULL.
12503	The VEH_ATTR_EXTERNAL_NAME should not be NULL.
12504	The UDV_EXTERNAL_NAME should not be NULL.
12505	The PROMO_EXTERNAL_NAME is not valid. The correct promotion should be included in the current data feed.
12506	The PROMO_OFFER_EXTERNAL_NAME is not valid.
12507	The VEH_ATTR_EXTERNAL_NAME is not valid.
12508	The UDV_EXTERNAL_NAME is not valid.
12509	The unrelated attribute values (VALUE_INT, VALUE_DEV, or UDV_ EXTERNAL_NAME) should be NULL. See the TYPE_ENUM field.
12510	The PROMO_OFFER_EXTERNAL_NAME is not valid. The correct value should be included in the current data feed.
APE Elasticity Loader Errors	
12701	DRIVER_APE_MERCH_NODE_EXT_ID should not be NULL.
12702	TARGET_APE_MERCH_NODE_EXT_ID should not be NULL.
12703	LOC_LEVEL_DESC should not be NULL.
12704	LOC_CLIENT_LOAD_ID should not be NULL.
12705	ELASTICITY should not be NULL.
12706	DRIVER_APE_MERCH_NODE_EXT_ID is not valid.
12707	TARGET_APE_MERCH_NODE_EXT_ID is not valid.
12708	Location is not valid.
12709	Vehicle Attribute is not valid.
Future Price Cost Changes Errors	
12720	MERCH_CLIENT_LOAD_ID should not be NULL.
12721	MERCH_LEVEL_DESC should not be NULL.
12722	LOC_CLIENT_LOAD_ID should not be NULL
12723	LOC_LEVEL_DESC should not be NULL.
12724	EFFECTIVE_DT should not be NULL.
12725	PRICE should not be NULL.
12726	COST should not be NULL.
12727	Merchandise is not valid.
12728	Location is not valid.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
Like Merchandise and Like Location Errors	
12740	MERCH_CLIENT_LOAD_ID should not be NULL.
12741	MERCH_LEVEL_DESC should not be NULL.
12742	LIKE_MERCH_CLIENT_LOAD_ID should not be NULL.
12743	LIKE_MERCH_LEVEL_DESC should not be NULL.
12744	MERCH_CLIENT_LOAD_ID and MERCH_LEVEL_DESC combination is not valid.
12745	LIKE_MERCH_CLIENT_LOAD_ID and LIKE_MERCH_LEVEL_ DESC combination is not valid.
12746	LOC_CLIENT_LOAD_ID should not be NULL.
12747	LOC_LEVEL_DESC should not be NULL.
12748	LIKE_LOC_CLIENT_LOAD_ID should not be NULL.
12749	LIKE_LOC_LEVEL_DESC should not be NULL.
12750	LOC_CLIENT_LOAD_ID and LOC_LEVEL_DESC combination is not valid.
12751	LIKE_LOC_CLIENT_LOAD_ID and LIKE_LOC_LEVEL_DESC combination is not valid.
12752	The record is part of a circular reference with another record.
12753	The value for INACTIVE is invalid. It cannot be NULL; it must be either 0 or 1.
12754	An active like item cannot be loaded if there is already an active record with the reverse relationship.
MB Detail Errors	
12760	TXN_ID should not be NULL.
12761	TXN_DATE should not be NULL.
12762	LOC_CLIENT_LOAD_ID should not be NULL.
12763	MERCH_CLIENT_LOAD_ID should not be NULL.
12764	Merchandise is not valid.
12765	Location is not valid.
12766	TXN_DATE is not a valid calendar period.
12767	The Units Sold value should not be NULL or zero.
SKU Lists Errors	
12901	NAME should not be NULL.
12902	EXTERNAL_NAME should not be NULL.
SKU List Items Errors	
12910	SKU_LIST_EXTERNAL_NAME should not be NULL.
12911	MERCH_CLIENT_LOAD_ID should not be NULL.
12912	MERCH_CLIENT_LOAD_ID is not a valid SKU.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
12913	SKU_LIST_EXTERNAL_NAME is not specified in the current import.
12914	SKU_LIST_EXTERNAL_NAME is invalid. (See corresponding SKU list BAD table.)
Store Sets Errors	
12921	STORE_SET_NAME should not be NULL.
12922	STORE_SET_DESC should not be NULL.
12923	INACTIVE_FLG should have a value of either 0 or 1.
12924	STORE_SET_TYPE should have a value of either 0 or 1.
12925	FIRST_EFF_DT should not be NULL.
12926	Too many new records have the same STORE_SET_NAME.
12927	The value for OLD_STORE_SET_NAME should have a value that corresponds to the value for STORE_SETS_TBL.STORE_SET_NAME.
Store Subset Errors	
12928	Too many records have the same STORE_SUBSET_NAME.
12931	STORE_SUBSET_NAME should not be NULL.
12932	STORE_SUBSET_DESC should not be NULL.
12933	INACTIVE should be either 0 or 1.
12934	ORDER_SEQ should not be NULL.
12936	The value for STORE_SET_NAME should have a value that corresponds to the value in STORE_SET_TBL.STORE_SET_NAME.
12937	The value for OLD_STORE_SUBSET_NAME should have a value that corresponds to the value for STORE_SUBSETS_TBL.STORE_SUBSET_NAME.
Store Subset Assignment Errors	
12940	LOC_CLIENT_LOAD_ID should not be NULL.
12941	LEVEL_DESC should not be NULL.
12942	NEW_STORE_SUBSET_NAME should not be NULL.
12943	NEW_STORE_SET_NAME should not be NULL.
12944	The value for LOC_CLIENT_LOAD_ID should have a value that corresponds to the value in LOCATION_HIERARCHY_TBL.CLIENT_LOAD_ID.
12945	The value for LEVEL_DESC should have a value that corresponds to the value in LOCATION_HIERARCHY_TBL.LEVEL_DESC.
12946	The value for STORE_SUBSET_NAME should have a value that corresponds to the value in STORE_SUBSETS_TBL.STORE_SUBSET_NAME.
12947	The value for STORE_SET_NAME should have a value that corresponds to the value in STORE_SETS_TBL.STORE_SET_NAME.
12948	LOC_CLIENT_LOAD_ID cannot be assigned to two subsets within a set.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
Predicted Baseline Errors	
12960	PERIOD_BEGIN is not a valid period start date.
12961	MERCHANDISE_ID should not be NULL.
12962	LOCATION_ID should not be NULL.
12963	MERCHANDISE_ID is not a valid merchandise.
12964	LOCATION_ID is not a valid location.
12965	BP_SUBST_MERCHANDISE_ID is not a valid merchandise.
12966	BP_SUBST_LOCATION_ID is not a valid location.
Promo Offer Criteria Errors	
13500	The value for EXTERNAL_NAME should not be NULL.
13501	The value for PROMO_EXTERNAL_NAME should not be NULL.
13502	The value for OFFER_EXTERNAL_NAME should not be NULL.
13503	The value for INCLUDE should not be NULL.
13504	The value for CRITERION_TYPE should not be NULL.
13505	The value for CRITERION_TYPE is not valid.
13506	The value for SKU_LIST_EXTERNAL_NAME should not be NULL.
13507	The value for SKU_LIST_EXTERNAL_NAME is not valid.
13508	The value for MERCH_CLIENT_LOAD_ID should not be NULL.
	The value for MERCH_CLIENT_LOAD_ID
13510	The value for ATTRIBUTE_NAME is not valid.
	The value for LEVEL_DESC should not be NULL.
	The value for EXTERNAL_NAME is not valid.
13513	The value for PROMO_EXTERNAL_NAME is not valid.The correct promotion should be included in the current data feed.
13514	The value for PROMO_OFFER_EXTERNAL_NAME is not valid.The correct promotion should be included in the current data feed.
13515	The value for ATTRIBUTE_VALUE should not be NULL.
13516	The value for ATTRIBUTE_NAME should not be NULL.
13517	The value for LOGICAL_OPERATOR is not valid.
13518	The value for ATTRIBUTE_VALUE is not valid.
13519	A SKU cannot be inserted if it is inactive.
13520	A SKU List cannot be inserted if it is inactive.
13521	A Category cannot be inserted if it is inactive.
APE Baseline Map Errors	
13600	The value for LOC_CLIENT_LOAD_ID should not be NULL.
13601	The value for LOC_LEVEL_DESC should not be NULL.
13602	The value for HISTORICAL_DATE should not be NULL.
13603	The value for PREDICTED_DATE should not be NULL.

Table 4–7 (Cont.) Standard Load Error Messages

Number	Error Message
13604	The value for LOC_CLIENT_LOAD_ID must have a corresponding value in LOCATION_HIERARCHY_TBL.CLIENT_LOAD_ID.
13605	The value for LOC_LEVEL_DESC must have a corresponding value in LOCATION_HIERARCHY_TBL.LEVEL_DESC.
13606	The value for HISTORICAL_DATA must have a corresponding value in PERIOD_TBL.BEGIN_CALENDAR_DT.
13607	The value for PREDICTED_DATE must have a corresponding value in PERIOD_TBL.BEGIN_CALENDAR_DT.
Baseline Dataset Errors	
13700	The value for RUN_ID should not be NULL.
13701	The value for PERIOD_ID should not be NULL.
13702	The value for MERCHANDISE_ID should not be NULL.
13703	The value for LOCATION_ID should not be NULL.
13704	The merchandise for MERCHANDISE_ID is not a valid merchandise.
13705	The location for LOCATION_ID is not a valid location.
ARM Pull Metrics Errors	
13800	The period dates do not match the beginning and end dates for the fiscal week.
Promote Periods Attribute Errors	
13900	The Dark Period Flag is invalid; it should be either N or Y.
13901	The value for the Begin Calendar Date should not be NULL.
13902	The value for the End Calendar Date should not be NULL.
Store Set Pricing Validation Errors	
14000	The value for MERCH_CLIENT_LOAD_ID should not be NULL.
14001	The value for MERCH_LEVEL_DESC should not be NULL.
14002	The value for STORE_SET_NAME should not be NULL.
14003	The value for STORE_SUBSET_NAME should not be NULL.
14004	The value for PRICE should not be NULL.
14005	The value for COST should not be NULL.
14006	The value for MERCH_LEVEL_DESC must have a corresponding value in CLIENT_HIERARCHY_LEVELS_TBL.LEVEL_DESC.
14007	The value for MERCH_CLIENT_LOAD_ID must have a corresponding value in MERCHANDISE_HIERARCHY_ TBL.CLIENT_LOAD_ID.
14008	The value for STORE_SET_NAME must have a corresponding value in STORE_SET_TBL.STORE_SET_NAME.
14009	The value for STORE_SUBSET_NAME must have a corresponding value in STORE_SUBSET_TBL.STORE_SUBSET_NAME.

Standard Load Procedures Order

The standard load should execute in the following order:

1. The staging of:
   - ASH_CAL_TBL
   - BEE_PERIODS_ATTR_TBL
   - ASH_CP_TBL
   - ASH_ITEMS_TBL
   - ASH_LHL_TBL
   - ASH_MHL_TBL
   - ASH_LHRENAME_TBL
   - ASH_LH_TBL
   - ASH_MHRENAME_TBL
   - ASH_MH_TBL
   - BEE_IMAGE
   - BEE_OFFER
   - BEE_USER_DEFINED_TYPE
   - BEE_USER_DEFINED_VALUE
   - BEE_VEHICLE_ATTR
   - BEE_VEHICLE
   - CLIENT_HIERARCHY_ACTIONS_TBL
   - STAGE_MH_ATTRS_TBL
   - BEE_STORE_SETS
   - BEE_STORE_SUBSETS
   - BEE_STORE_SUBSET_ASGN
   - BEE_PR_LIKE_LOCATION
   - BEE_PR_LIKE_MERCHANDISE
   - BEE_SKU_LIST
   - BEE_SKU_LIST_ITEM
   - BEE_PROMO_ALLOC
   - BEE_PROMO_CAMPAIGN
   - BEE_PROMO_OFFER_ATTR
   - BEE_PROMO_OFFER_MERCH
   - BEE_PROMO_OFFER
   - BEE_PROMO_STORE
   - BEE_PROMOTIONS
   - ASH_PARAMETER_VALUES_TBL
   - ASH_SEASONALITY_MAPS_TBL

- ASH_SEASONALITY_VALUES_TBL
- BEE_FUTURE_PRICE_COST
- BEE_MB_DETAIL

2. These load procedures:
   - com.profitlogic.db.birch.LoadCalendars
   - com.profitlogic.db.birch.LoadCPLevels
   - com.profitlogic.db.birch.LoadLHKeyRename
   - com.profitlogic.db.birch.LoadMHKeyRename
   - com.profitlogic.db.birch.LoadMerchandiseHierarchy
   - com.profitlogic.db.birch.LoadLocationHierarchy
   - com.profitlogic.db.birch.LoadMHTbl
   - com.profitlogic.db.birch.LoadLHTbl
   - com.profitlogic.db.birch.LoadTClose
   - com.profitlogic.db.birch.LoadLTClose
   - com.profitlogic.db.beech.LoadTypeMaster
   - com.profitlogic.db.beech.LoadValueMaster
   - com.profitlogic.db.beech.LoadOfferMaster
   - com.profitlogic.db.beech.LoadImageMaster
   - com.profitlogic.db.beech.LoadVehicleMaster
   - com.profitlogic.db.beech.LoadVehicleAttributeMaster
   - com.profitlogic.db.birch.LoadMerchandiseAttributes
   - com.profitlogic.db.beech.LoadPromoLikeLocation
   - com.profitlogic.db.beech.LoadPromoLikeMerchandise
   - com.profitlogic.db.beech.LoadSkuListMaster
   - com.profitlogic.db.beech.LoadSkuListItemMaster
   - com.profitlogic.db.beech.LoadSkuListCleanupMaster
   - com.profitlogic.db.beech.LoadStoreSets
   - com.profitlogic.db.beech.LoadStoreSubsets
   - com.profitlogic.db.beech.LoadStoreSubsetAssignments
   - com.profitlogic.db.beech.LoadPromoCampaign
   - com.profitlogic.db.beech.LoadPromoMaster
   - com.profitlogic.db.beech.LoadPromoVehicle
   - com.profitlogic.db.beech.LoadPromoVehiclePage
   - com.profitlogic.db.beech.LoadPromoVehicleAlloc
   - com.profitlogic.db.beech.LoadPromoVehicleLocation
   - com.profitlogic.db.beech.LoadPromoOffer
   - com.profitlogic.db.beech.LoadPromoOfferMerchandise

- com.profitlogic.db.beech.LoadPromoVehiclePagePosOff
- com.profitlogic.db.beech.LoadPromoVehiclePagePositionOfferAttribute
- com.profitlogic.db.birch.LoadParameters
- com.profitlogic.db.birch.LoadSeasonalities
- com.profitlogic.db.beech.LoadFuturePriceCost
- com.profitlogic.db.walnut.CreateMerchSummaryTbl
- com.profitlogic.db.walnut.CreatemerchPromoMBSummaryTbl
- com.profitlogic.db.walnut.CreateMerchMapView
- com.profitlogic.db.beech.LoadMbDetail

3. Staging WK_HIST_SALES_INV
4. Loading load_weekly_history_data.load_at_all_levels
5. pr_load_promo_mb_summary.load_all
6. pr_create_store_count_sum

Standard Load Steps

Each procedure consists of the following sub-procedures:

1. Setup
2. Pre-load Verification. All n processes are run in parallel.
3. Finish Pre-load Verification.
4. Load. All n processes are run in parallel.
5. Post-load Verification. All n processes are run in parallel.
6. Finish Post-load Verification.
7. Tear-down.

Standard Interface Specifications for One-Time Data

The following three standard interface specifications are used for data that is loaded once at the beginning of a Promote deployment.

Cross Products Information Standard Interface (ASH_CP_TBL)

Items are globally defined to be at a specific level of the merchandise hierarchy and the location hierarchy through the cross products interface.

Technical Notes

The following list provides details to considering regarding the cross products information data.

- The INTERSECT_NAME is the name of the Key, which identifies the purpose or feature for the data.

  The value is either OPTIMIZATION, SALES, WORKSHEET, CLUSTER, PROMOTE_PROMO_OFFER_MH_SUMMARY, PROMOTE_SCORECARD_MERCH_OFF_AMT_SUMM_3, PROMOTE_SCORECARD_MERCH_OFF_AMT_SUMM_2, PROMOTE_SCORECARD_MERCH_OFF_AMT_SUMM_1, or

DEFAULT LEVEL. The PROMOTE_PROMO_OFFER_MH_SUMMARY value specifies the aggregation level for the merchandise hierarchy used to generate scorecard totals in reports. The PROMOTE_SCORECARD_MERCH_OFF_AMT_SUMM_*n* entries are used to generate the totals for the scorecard by merchandise hierarchy and offer amount.

- For each Key, identify the defining level of the merchandise hierarchy and location hierarchy.
- The cross products information is generally loaded only once.

Cross Products Information Specification

Table 4–8 Cross Products Information Standard Interface Specification

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
INTERSECT_NAME	See Table 4–9, " Promote Cross Product Information Intersect Names" for details about each intersect name.	String	50	N
MERCHANDISE_LEVEL	The defining level within the hierarchy	String	50	N
LOCATION_LEVEL	The defining level within the hierarchy	String	50	N

Cross Product Information (ASH_CP_TBL) Intersect Names

Promote supports the following intersect names.

Table 4–9 Promote Cross Product Information Intersect Names

Intersect Name	Used To...
DEFAULT	
OPTIMIZATION	
SALES	
WORKSHEET	
PROMOTE_PROMO_OFFER_MH_SUMMARY	Specifies the level of aggregation from the Merchandise Hierarchy that is used to generate the totals for the Scorecard by Offer/Dept report.
PROMOTE_TAE_NONAD_PART_LEVEL_*n*	Specifies the level of aggregation from the Merchandise Hierarchy that is used for the TAE Non-Ad metrics. It is also used by the TAE process to identify the starting Merchandise Hierarchy level that should be used to generate its output.
PROMOTE_SUMMARY_*n*	
PROMOTE_SCORECARD_SUMMARY_*n*	Specify the level of aggregation from the Merchandise Hierarchy that is used to generate the totals for the scorecard by the Merchandise Hierarchy.

Table 4–9 (Cont.) Promote Cross Product Information Intersect Names

Intersect Name	Used To...
PROMOTE_SCORECARD_MERCH_OFF_AMT_SUMM_*n*	Specify the level of aggregation from the Merchandise Hierarchy that is used to generate the totals for the Scorecard by Merchandise Hierarchy and Offer Amt.
PROMOTE_SCORECARD_TOP_NONAD	Used by the load process to select only the specified top N contributors from staging to target tables.
PROMOTE_MIN_LCD	Defines the lowest level of the hierarchy that is available for display in the UI.
PROMOTE_MIN_BL_AGGR_LEVEL	Specifies the lowest level that aggregated baseline data should be calculated for.
PROMOTE_MAX_BL_AGGR_LEVEL	Specifies the highest level that aggregated baseline data should be calculated for.
PROMOTE_TAE	Identifies the level at which TAE output is produced.
PROMOTE_DETAIL	Identifies the level at which POS data is expected. It is assumed to be the STORE level.
PROMOTE_AFFINITY_LEVEL	The level of calculation of the APE summary.
PROMOTE_APC	The level of calculation of the APC summary.
PROMOTE_ANALYSIS	

Location Hierarchy Levels Standard Interface (ASH_LHL_TBL)

The location hierarchy levels interface is used to specify the names of a retailer's location levels and their order.

Technical Notes

The following list provides details to consider regarding the lh levels data.

- The Chain level should always be defined as 1.
- The sequence of level numbers must begin with 1 and increase in increments of 1, without any gaps in the sequence.
- The location hierarchy levels information is generally loaded only once.

LH Levels Specification

Table 4–10 Location Hierarchy Levels Standard Interface Specification

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
LOCATION_LEVEL	The name of the location level	String	50	N
LEVEL_SQC	The sequence number of the level	Integer	2	N

Merchandise Hierarchy Levels Standard Interface (ASH_MHL_TBL)

The merchandise hierarchy levels interface is used to specify the names of a retailer's merchandise levels and their order.

Technical Notes

The following list provides details to consider regarding the mh levels data.

- The Chain level should always be defined as 1.
- The sequence of level numbers must begin with 1 and increase in increments of 1, without any gaps in the sequence.
- The merchandise hierarchy levels information is generally loaded only once.

MH Levels Specification

Table 4–11 Merchandise Hierarchy Levels Standard Interface Specification

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_LEVEL	The name of the merchandise level	String	50	N
LEVEL_SQC	The sequence number of the merchandise level	Integer	2	N

5

Historical Analysis

This chapter contains the following:

Introduction

Historical analysis consists of baseline analysis followed by tae analysis. Baseline analysis defines the baseline behavior of a product absent promotion activity. Tae analysis defines the lift provided by a promotion. This chapter provides operational details about historical analysis.

Baseline Analysis

Baseline metrics are the measurements of the behavior of an item during the time that it is not being promoted. These metrics are used as the standard of comparison in order to determine the effect of the promotion on the item. The difference between the baseline behavior of an item and that item's behavior during a promotion is called the *lift*.

The following metrics are measured by the baseline utility:

- Rate of sale (ROS)
- Visit rate (VR)
- Gross sales
- Gross margin
- Baseline mode price

Baseline metrics (sums and averages) are measured per store per item per period (typically a week). The baseline application generates baseline measurements only for item/store pairs that have at least one day of non-promotion sales and no promotion sales for a given period.

Running Baseline

The baseline utility outputs the results to a database table using the following scripts:

- pl_stage_promote.sh <configuration file location> <control file location> <dataset name and location>
- pl_load_baseline <configuration file location>

The baseline utility has the following options:

baseline.sh -out <*dataset name or location*> **-startdate** <*yyyy-mm-dd*> **-enddate** <*yyyy-mm-dd*> **[-append] [-kdehelp] [-help]**

The **-out** option accepts the name of the dataset file, such as baseline.ds, or the full pathname to the dataset.

The **-startdate** option should be the first day (which is a Sunday) of the first period being measured.

The **-enddate** option should be the last day (which is a Saturday) of the last period being measured.

The **-append** argument is optional. If this argument is included, the results of each successive run of the utility are appended to the dataset. If the output dataset exists and this option is not specified, then an error occurs. If the output dataset has not been created and this option is specified, there is no effect.

The **-kdehelp** option displays the full KDE command syntax. For information about the command options for kde.sh baseline, see Chapter 10, "Technical Reference".

Best Practices

For best analytical results, the following practices are recommended:

- Run the baseline utility and output the data to a temporary dataset instead of a production dataset. This allows you to review the results before committing changes to the production dataset.
- Process large amounts of transactional data with kde baseline in separate batches. for example, process the data in monthly or weekly increments.
- After the initial execution of kde baseline, use the -append option with every subsequent execution of the utility in order to append the metrics produced to a single history data set.

TAE Analysis

TAE analysis measures the effectiveness of promotions, based on the baseline data generated by kde baseline. The output of Tae is a Promotion Scorecard, containing metrics and statistics for Focus Items (items that were part of a promotion).

TAE calculates the following metrics for specific focus items and the overall ad performance:

- The average rate of sale, visit rate, and mode price for a focus item during one or more baseline periods.
- The total rate of sale, visit rate, and price for a focus item during a promotion. Incremental metrics can be derived by subtracting the baseline metrics from the ad-period metrics.

- The amount of drag or pull of non-ad items associated with or allocated to a focus item.
- The incremental and cross-sell effects of a focus item on other items.
- The total effect of the ad relative to the incremental ad-item and non-ad-item rate of sale, visit rate, revenue, and profit margin.
- Comparisons with similar ad events.

Running TAE

The tae utility outputs the results to a database table specified by the variables $TAE_TEMP_METRIC_TABLE and $DB_VAR. The utility has the following options:

tae.sh -adenameid <external_id> [-algorithm <algorithm>**] [-basewindow** <yyyy-MM-dd,yyyy-MM-dd,pmin>**] [-inhistory** <location of historical baseline dataset>**] [-kdehelp] [-help]**

Note also the tae_dbl.sh utility, which is similar to tae.sh, but without the -inhistory option, and which is used to calculate some TAE baseline metrics dynamically.

tae_dbl.sh -adenameid <external_id> [-algorithm <algorithm>**] [-basewindow** <yyyy-MM-dd,yyyy-MM-dd,pmin>] **[-kdehelp] [-help]**

Options for tae.sh and tae_dbl.sh

The **-adnameid** option specifies the external ID of the ad event being measured. Only one event can be analyzed by TAE at a time.

The **-algorithm** option indicates which baseline period should be used for comparison purposes. This option supports one of the following:

- **py** – a baseline window one year prior to the current date is used.

  The number of weeks in the window is specified in the kde.properties file in com.netperceptions.kde.prior.year.weeks. The default value is 3.

  The minimum number of periods in the window is specified in the kde.properties file in com.netperceptions.kde.prior.year.pmin. the default value is 2.

- **baa** – baseline windows before and after the ad period are used. The size of each baseline window is configured separately. It is thus possible to set one of the weeks (before or after) to a value of 0 in order to provide only one window.

  The window settings are specified in the kde.properties file, as follows.

  – com.netperceptions.kde.baa.before.weeks - the number of weeks before the promotion to be used in the base window. The default value is 3.

  – com.netperceptions.kde.baa.before.pmin - the minimum number of periods before the promotion that can be used in the base window. The default value is 2.

  – com.netperceptions.kde.baa.after.weeks - the number of weeks after the promotion to be used in the base window. The default value is 3.

  – com.netperceptions.kde.baa.after.pmin - the minimum number of periods after the promotion that can be used in the base window. The default value is 2.

- **cafe** – both py and baa are used.

The **-basewindow** option can be used to define the baseline period for comparison purposes using specific start and end dates.

- **start** – the date of the first day of the first period to be used.
- **end** – the date of the last day of the last period to be used.
- **pmin** – the minimum number of periods in which the item/location combination must be found in the history data. The value for pmin must be between the number of periods specified in the window and 1.

The **-inhistory option** is not required. If not used, then tae.sh uses the value of the variable BASELINE_OUT_DS that is set in kde.var for its default value.

The **-kdehelp** option displays the full KDE command syntax. For information about the command options for kde.sh tae, see Chapter 10, "Technical Reference".

Note the following:

- Specifying a promotion inside the window causes an error.
- More than one -basewindow can be specified on the command line.
- Either -algorithm <algorithm_id> or -basewindow <start,end,pmin> is required; however, both can be specified.
- The default, production, output dataset should be called tae.metric.ds. This can be changed in the kde.vars configuration file.

Some Examples

Here are two examples of using tae.sh.

The first example analyzes promotion 0199-A using baseline data from a period beginning three weeks before the ad event and ending three weeks after the ad event. The results are stored in a temporary tae dataset.

tae.sh -adnameid 0199-A **-algorithm** baa

The second example measures the same event, but uses a different baseline period from the previous year.

tae.sh -adeventid 0199-A **-basewindow** 2006-08-20,2006-09-09,2 **-basewindow** 2006-09-17,2006-10-07,2

The output of these two examples can be compared to determine the best baseline approach for this ad event using reports, as discussed in the next section.

Uploading the Database

Certain reports depend on the TAE data to function. The following commands are provided to facilitate the use of these reports and the analysis of the TAE results. These commands copy the TAE data to the database.

This command stages the data to the dataset. If no dataset is provided, the default dataset (defined in kde.vars) is used.

tae_stage.sh [-in <*dataset name or location*>**] [-help]**

This command loads the previously staged data into the production data warehouse.

tae_load.sh [-help]

This command stages the data to the dataset. The dataset name or path is required.

tae_stage_temp.sh -in <*dataset name or location*> **[-help]**

This command loads the previously staged data into the temporary reporting tables.

tae_load_temp.sh [-help]

TAE Results

The output of the TAE analysis is a Promotion Scorecard. The following metrics are included in the TAE Scorecard Report.

Ad Item Metrics are for ad items only.

- **ad_item_price** – the average price of the ad item
- **ad_item_rosale** – the sum of the item
- **ad_item_visit_rate** – the visit rate or the number of market baskets that the item was in
- **ad_item_sales** – the total sales amount of the ad- item
- **ad_item_gm** – the total gross margin amount of the ad-item
- **ttl_ad_days** – the number of days in the ad period that the item actually sold

Affinity Correlation

Note that these values will be zero if the ac_level is not specified in the MB_Detail data.

- **ad_item_ac_gm** – Affinity average gross margin
- **ad_item_ac_sales** – Affinity average sales
- **ad_item_ac_units** – Affinity average sold units
- **ad_item_pr_gm** – Revenue average gross margin
- **ad_item_pr_sales** – Revenue average sales
- **ad_item_pr_units** – Revenue average sold units
- **ad_nonad_gm** – Non-ad average gross margin
- **ad_nonad_sales** – Non-ad average sales
- **ad_nonad_units** – Non-ad average sold units

These items show the status of baseline data being used.

- **bl_subst_code** – the substitution code
- **bl_subst_item** – the item used if substitution is used

Where the values for bl_subst_code are:

- 0 – the focus item has enough baseline
- 1 – the focus does not have enough baseline, but was an item was substituted that does have enough baseline.
- 2 – the focus item does not have enough baseline and substitution is turned off.

- 3 – the focus item does not have enough baseline and a substitution item was not available.
- 4 – the focus item does not have enough baseline and the substitution item does not have enough baseline either.

If substitution is turned on, the TAE will classify the focus items into either class 0, 1, 4, or 5. Focus items that have enough baseline go into class 0. The others are joined with the substitution dataset. Any records without substitution matches go into class 3. Successful substitutions are then checked to ensure they have enough baseline. Those that do go into class 1. The others go into class 4.

If substitution is turned off, only class 0 and 2 are used. Those focus items with enough baseline go into class 0. Those without go into class 2.

These are the averages for the periods in the chosen baseline window for the item or substitute item. Theses items are all zero if the bl_subst_code is 2, 3 or 4 as explained below.

- **ttl_base_periods** – total number of periods in the baseline window
- **bl_item_rosale** – average rate of sale for the item in the baseline window
- **bl_item_sales** – average sales for the item in the baseline window.
- **bl_item_visit_rate** – average visit rate or market basket count for the item in the baseline window.
- **bl_item_gm** – average gross margin for the item in the baseline window
- **bl_item_price** – average price for the item in the baseline window.

These are the same as the ad-items explained above with exception that these were generated from the MB Detail data from the selected baseline window. These values are included when the option "blalloc" is specified. These values will be zero if the ac_ level is not specified in the MB Detail data.

- **bl_item_ac_gm**
- **bl_item_ac_sales**
- **bl_item_ac_units**
- **bl_item_pr_gm**
- **bl_item_pr_sales**
- **bl_item_pr_units**
- **bl_nonad_gm**
- **bl_nonad_sales**
- **bl_nonad_units**

These are the quality metrics. These values will be zero unless the option *quality* is specified. They really seem to be market basket analysis instead of quality.

- **ad_mb_item_only** – the number of promotion baskets that contain ad item alone
- **ad_mb_item_ad** – the number of promotion baskets that contain ad item and some other ad items, but no nonad items
- **ad_mb_item_nonad** – the number of promotion baskets that contain ad item and at least one nonad item, but no ad items other than the ad item

- **ad_mb_item_adnonad** – the number of promotion baskets that contain ad item, other ad items, and at least one nonad item
- **ad_item_othad_ros –** the count of the remaining item in the market basket

Substitution

The kde tae utility requires baseline data generated by the kde baseline utility. Baseline data may not be available because the item is new or because the item is frequently on promotion at the given location.

If no baseline data is available, the kde tae utility generates a scorecard with zero baseline numbers and flags the bl subst code column. (See "TAE Results" on page 5-5).

If a focus item does not have enough sales history to calculate the baseline, it is possible to use the behavior of a like item to calculate the baseline for the new item. It is also possible to define how much data is considered reliable for calculating the baseline. Adequate baseline parameters are based on the count of the number of uncontaminated baseline periods that occur in each baseline window.

To assign a like item to another item that does not have adequate sales data, create a mapping table that maps new items to old items that have similar sales behavior. TAE substitutes the baseline data from the old item. When baseline substitution occurs for a store/item combination, the corresponding TAE output contains a flag that indicates that the substitution occurs and a field specifies the substituted item.

Like Item Mapping

Like items can be configured using either the application user interface or via the data feeds. For more information, see the *Promotion Planning and Optimization User Guide* or Chapter 3, "Standard Interface".

Best Practices

For the best analytical results, the following practices are recommended:

- The value for the affinity index threshold (the -minlift option) should be 2 - 5. A higher value finds fewer affinities, but with a greater degree of confidence.
- When running the baseline utility, output the data to a temporary dataset instead of a production dataset. Results can then be reviewed before the production dataset is modified.

ARM Analysis

Association Rule Mining (ARM) is the technique used by the analytics to compute a variety of association metrics. These metrics describe how promoting one product affects the sales of other products in the historical dataset.

Two different analysis methods can be performed using ARM. To perform an analysis most effectively, you should use both methods.

The first method performs the analysis weekly, irrespective of whether or not an item has been promoted. The results of this method are available in the Affinity report.

The second method, the ARM Pull, also performs the analysis weekly; however this method analyzes the relationship between items while it tracks whether or not the items have been promoted.

At intervals defined by the needs of the business, the results of the weekly ARM analysis is summarized. This analysis is used to detect patterns in the results that indicate which items pulls along the sales of another item. These results are available in the Affinity Pull report.

1. To use the first method to run ARM weekly, do the following:

```
$cd ${PCE_HOME}/bin
$arm.sh -minconfidence 0.05 -runid 1 -maxsetsize 2 -startdate YYYY-MM-DD
-enddate YYYY-MM-DD
```

2. Once the ARM data has been computed, it can be loaded into the database (RDM) for further analysis and reporting.

```
$ cd ${PCE_HOME}/bin
$ arm_stage.sh
$ arm_load.sh
$ cd ${installdir}/modules/Database/SEQUOIASchema/install/oracle/SEQUOIASchema
/scripts
$ ./pl_refresh_ARM_sets.sh ${installdir}/config rdm_plexports.sh
```

3. To use the second method to run ARM Pull weekly, do the following:

```
$ cd ${PCE_HOME}/bin
$arm_pull.sh -minconfidence 0.05 -startdate YYYY-MM-DD -enddate YYYY-MM-DD
```

   Once the ARM Pull data has been computed, you can save it to the database for later analysis:

```
$ cd ${PCE_HOME}/bin
$ arm_pull_stage.sh
$ arm_pull_load.sh
```

Either quarterly, or at another appropriate time, you should run the following scripts in order to summarize the weekly ARM results and make them available for reporting. You should use the same dates for each of the two scripts. You provide two sets of dates for each script; the first date is the start date and the second date is the end date for the time period you are analyzing.

To perform an ARM summary analysis, do the following:

```
$ cd ${installdir}/modules/Database/WALNUTSchema/install/oracle/WALNUTSchema
/scripts
$ pl_summarize_arm_runs.sh ${installdir}/config promote_plexports.sh
YYYY-MM-DD YYYY-MM-DD
```

To perform ARM Pull summary analysis, do the following:

```
$ cd ${installdir}/modules/Database/WALNUTSchema/install/oracle/WALNUTSchema
/scripts
$ pl_create_arm_pull_summary.sh ${installdir}/config promote_plexports.sh
YYYY-MM-DD YYYY-MM-DD
```

Once you have generated the ARM summary data, you can publish it to the RDM to be used for reporting, as follows:

```
$ cd ${installdir}/modules/Database/SEQUOIASchema/install/oracle/SEQUOIASchema
/scripts
$ ./pl_refresh_ARM_sets.sh ${installdir}/config rdm_plexports.sh
```

Market Basket Analysis

Promote includes tools to use in order to both perform market-basket analysis of client data and populate the Retail Data Mart (RDM) with information about ad effectiveness. The data in the RDM is customized and aggregated so that it can be easily accessed for reporting. Note that the RDM cannot be set up until the POS data has been loaded into the Promote database.

Analysis

Complete the following steps to run the market basket analysis:

1. Configure the following parameters, which define the aggregation levels that are created during the market basket analysis:
   - PROMOTE_SUMMARY_*n*
   - PROMOTE_SCORECARD_SUMMARY_*n*
   - PROMOTE_PROMO_OFFER_MH_SUMMARY
   - PROMOTE_SCORECARD_MERCH_OFF_AMT_SUMM_*n*
   - PROMOTE_TAE_NONAD_PART_LEVEL_*n*
2. Run the script **pl_create_summary_tables.sh, pl_create_offer_amt_summ_tables.sh, and pl_merch_promo_merch_summ_tbls.sh**. These scripts creates the database tables required to store the market-basket summary data at different levels of aggregation.
3. To create the application's views, run the scripts **pl_create_map_views_sh** and **pl_create_promote_ir_views.sh**. These scripts create some customized views that are used during the market basket analysis.
4. Refresh static tables used for the market basket summary, using the latest promotion data. Run the scripts **pl_load_promoitem.sh** and **pl_load_promolocation.sh**.
5. Run the script **pl_create_summaries.sh**. This script creates summary market basket information at different levels of detail, as defined by the configuration parameters defined in step 1.
6. Run the script **pl_create_RDM_summary.sh**. This script creates views and synonyms on the RDM schema that are used by reports.
7. Run the script **pl_load_promote_RDM.sh**. This script copies over to the RDM basic promotion information, the merchandise and location hierarchies, and the periods data used by the entire application.

6

Predictive Modeling

This chapter contains the following:

Introduction

The Promote Calc Engine (PCE) analyzes historic trends, builds models for prediction, and performs analysis of What-if scenarios. The PCE consists of a set of tools, functionally organized as follows:

- Historical Analysis Module – provides tools to measure promotions that have occurred in the past.
- Baseline Prediction Module – provides tools to create a demand forecast, absent any promotions, for future time periods.
- Model Development Module – provides tools for creating, testing, and deploying customer-behavior models.
- Prediction Module – provides tools for real-time performance analysis.

The Historical Analysis Module is discussed in Chapter 5, "Historical Analysis". the other three modules are discussed in this chapter.

Forecasting is a feature that allows a user to interactively run what-if scenarios, based on the data provided.

The forecasting configuration points are analytical modeling files, database views, and database tables.

Building Models

The Model Development module includes tools for creating, testing, and deploying models of customer behavior. The tools include modeldataprep, modelbuild, adgenstats, and modeldeploy.

Adjusting the Model

If needed, adjust the model as follows:

- Configure Promote metadata
- Ensure that the data feeds have this data
- Update the model to add or adjust the attributes

The Promote Calc Engine models past promotion performance using a set of configurable attributes. To control the attributes used in the modeling process, edit the model_config.xml file located in <install-dir>/modules/pce/data/models/default. This file is also used to configure other options of the modeling process, such as the modeling algorithm, training time periods, minimum lift, and seed values.

modeldataprep

The modeldataprep analytic utility merges data from the model building source data sets into two data sets that are segmented by time. Typically, the earlier data is used for training and the later data is used for testing.

In addition to merging and time-segmenting the data, modeldataprep also synthesizes two attributes from the input data: item lift mean and seasonality scale factor.

- The item lift mean is a product attribute that indicates, generally, how the item responded to promotion in the past.
- The seasonality scale factor is a measure of how the product sold at each store-week of the year.

The utility also prunes and filters the data to avoid training on noise and statistically insignificant events.

modelbuild

The modelbuild analytic utility builds models using a number of predictive and clustering algorithms. A common property of these algorithms is that they take an input stream of training data and produce a model that describes the data and that is suitable for input to the modelapply application.

The modelbuild utility takes as input a mining schema, specified by the option -miningSchema, and produces a PMML (Predictive Modeling Make-up Language) model file.

adgenstats

The adgenstats utility uses the model test and training data to generate a chain-level stats file for use by modelapply, with a particular model, when aggregation is on. These chain-level statistics are not necessary if aggregation is off. The input to adgenstats is the training data from modelbuild. The output is a file that contains chain-level statistics. This file is used by modelapply to compute the statistics that are necessary to z-score the aggregates.

modeldeploy

The modeldeploy utility provides the test and training datasets to adgenstats. It also publishes the model information to the application database. The application database must include all the attributes that are referenced in the model.

Building Predictive Baselines

The following utilities are used for building predictive baselines.

adblcompute

This utility uses the baseline data to generate projected (future) baseline information for all sold items. It also performs checks of the start and end dates of the baseline window.

adblprep

Once the predicted baseline data is built (using adblcompute), the adblprep utility renames some columns and adds customer-specific attributes, such as the seasonality scale factor and liftmean data. The resulting baseline data segments are organized by specific model and stored in a directory, specified by "-bldir" attribute.

Using blcompute

Promote provides a means to define Like Items and Like Stores for new items and new stores that do not yet have sales data. The utility blcompute is used for Like Item substitution and Like Store substitution for prediction. Unlike tae substitution (for Like Items only), blcompute always performs substitutions when requested, so there is no feedback that substitution has occur for a given Focus Item or Focus Store.

Building Predictions

The modelapply utility is a what-if predictive tool and it is used by the Promotion Manager to perform real-time prediction of promotional performance. It applies an input model to an input dataset and outputs the results.

Building Predictive Baseline Process

Here is the process for building the predictive baseline (PBL). The baseline transaction data must first be analyzed by PCE and the results must be written to the database.

1. Define merchandise views. A typical use-case is to have 'basic' SKUs, which is selling all the time and would benefit from a 'sliding' baseline window, utilizing fresh data; and 'seasonal' SKUs, which exhibit behavior according to the calendar. PBL major parameters are the target future promotion week; view, that represents the merchandise; corresponding baseline window; switches (e.g. -useClearancePeriods -useDarkPeriods). For example, the views might be:
   - IR_PURCHTYPE_BASIC - SKUs, which require fresh baseline window.
   - IR_PURCHTYPE_BASIC_BAD - similar to a), but selling at lower rate and hence require bigger baseline window.
   - IR_PURCHTYPE_SEASONAL - SKUs, which require seasonal (e.g. 52-week-back) baseline window.
   - IR_PURCHTYPE_SEASONAL_BAD - similar to c), but selling at lower rate and hence require bigger baseline window.
2. Define baseline windows.
   1. For each PBL view, determine the rule that defines the window.

2. Populate PR_PBL_WINDOW table, using convenience functions (pr_mgmt.fill_*_pbl_windows).

3. Decide minor parameters (e.g. -useClearancePeriods).

4. Run PBL.

5. Define Offer Level Forecast (OLF) merchandise views: create IR_OLF_CANDIDATES_VW view that will return MERCHANDISE_ID of items which need to receive OLF (e.g. all the 'seasonal' SKUs) and IR_OLF_NODES_VW view that will define the aggregation level merchandise of OLF (e.g. all the merchandise nodes above SKU).

6. Define OLF parameters and baseline windows. Populate PR_PBL_MERCH_WINDOW table utilizing convenience functions (pr_mgmt.clean_aggr_pbl_windows / pr_mgmt.fill_aggr_pbl_windows), specifying the OLF view (IR_OLF_NODES_VW) and OLF parameters (alpha1/alpha2/discountMin/discountMax/windowSize/lookBack).

7. Run OLF.

8. Run the RG Metrics process for each of the views that were defined in Step 1 and used in Step 4.

7

Affinity Modeling

This chapter contains the following:

Introduction

This chapter describes the requirements for the affinity-modeling feature, which allows the product to produce predictions about the positive and negative sales impacts of planned promotions due to halo and cannibalization (substitution) effects.

The three types of affinity effects are:

- Complimentary halo effects of planned promotions
- Indirect (traffic) effects of planned promotions
- Cannibalization effects of planned promotions

Understanding the three types of affinity effects helps managers make better advertising decisions.

The Affinity Estimation Process

The Affinity Parameter Estimator (APE) is used by an analyst to discover, analyze, and refine the elasticity factors used during prediction. The APE includes a component that is responsible for estimating affinity effects. The APE is a separate application that is not included with the product CD.

The affinity process and the part the APE plays in that process is as follows:

1. The POS data is mined by the Promote Calc Engine (PCE) into aggregate form (store, item, week).
2. The aggregate data is saved to the database.
3. The analyst builds a configuration for the Promote Analytics module. This configuration details how to prepare the data for the APE. The analyst calls a process within the Promote Analytics module that performs the data preparation. The data is then saved to the database.

4. The analyst initializes the APE for the estimation of the affinity parameters.
5. The APE receives the previously prepared data from the application database and uses it to estimate the affinity parameters.
6. The APE provides the affinity parameters to the database.
7. The end user requests the predicted performance of an item for a planned promotion.
8. The Promotion Manager passes the request to the PCE.
9. Using the stored affinity parameters, the PCE computes the affinity information for the selected product.
10. The projected affinity effects are passed back to the Promotion Manager for display to the end user.

Choosing a Tool for APE Analysis

Any software package that supports basic statistical operations such as linear and non-linear modeling can be used as an APE. The instructions and examples in this guide assume that the "R" package (http://www.r-project.org/) is being used; however, any similar program such as S-Plus or SAS can also be used.

The tool used for APE analysis is not included with the product. However, the product CD does include some example scripts in the *<InstallationDirectory>*/modules/pce/ape directory. These example scripts are "R" compatible, but can be modified so that they can be used with other analytic packages.

APE Hierarchy

The APE uses a tree-like merchandise structure for performing predictions. In this way, it can identify affinity effects that exist at various levels above the item level. The APE uses its own independent hierarchy instead of the existing Promote hierarchical merchandise structure. This allows the user the flexibility to determine the best way to group items to suite their affinity effect prediction needs. While the existing Promote merchandise hierarchy has one root node, the APE hierarchy can have more than one, based on how the APE hierarchy has been configured. Affinity parameter estimation is only performed within an APE hierarchy tree, not across trees. Each item within the APE hierarchy belongs to one and only one hierarchy tree.

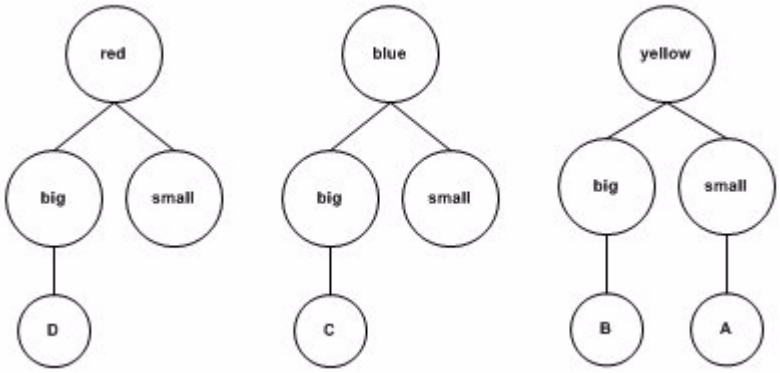
The APE hierarchy is built using merchandise attributes, such as size, color, or parent merchandise hierarchy node ID (from the existing hierarchy). To define a level in the APE hierarchy, such as the root level, a merchandise attribute or a set of attributes is specified. Under each node in the level above this level, a node is then created for every unique value of this attribute or for every unique combination of attribute values if more than one attribute is used. The set of attributes that can be used to define APE hierarchy levels is limited to the set that already exists within the client's existing merchandise data.

Once every level in the APE hierarchy is defined using merchandise attributes, the hierarchy can be built and items can then be assigned to their respective nodes. For example, if the first level in the APE hierarchy is defined by the attribute *color* and the set of unique values of this attribute are *blue*, *red*, and *yellow*, then three nodes are created at this level, one for each value of the color attribute. If the second level of the hierarchy is defined by the attribute *size* and the set of unique values for this attribute are *small* and *big*, then six nodes are created at this level, one set each of big and small nodes under the blue, red, and yellow nodes. Items whose color attribute have the

value *yellow* and whose size attribute have the value *small* would be assigned to the first level node yellow and the second level node small under the yellow node. Note that, in this scenario, a small node exists under several first level nodes. Which of these small nodes an item belongs to depends on which first level nodes the item belongs to.

Item	Color	Size
A	yellow	small
B	yellow	big
C	blue	big
D	red	big

Nodes:



The end result of this mapping is a new set of database tables that defines the APE merchandise hierarchy, mapping the item (SKU-level) IDs from the pre-existing merchandise hierarchy into their respective APE hierarchy nodes.

Hierarchy Tree Depth

Each APE hierarchy tree has three levels, 0-2, where the 0th level is the root level. In the simplest scenario, an APE hierarchy is created by defining one set of level zero (root level) merchandise attributes, one set of level one merchandise attributes, and one set of level two merchandise attributes. However, this "one size fits all" hierarchy may not an acceptable way to segment a client's entire merchandise set. Take for example the case of a client who wishes to define two APE hierarchy trees, one for beverages and one for shoes. With beverages, the client might be most interested in seeing the interactions between flavors. Style might be a more important trait with shoes. In this situation it would be useful to be able to define first level nodes for the beverage tree by a flavor attribute and to define first level nodes for the shoe tree by a size attribute.

Anchor Nodes

The use of anchor nodes is one way for the user to apply the above type of discrimination between sets of APE hierarchy trees. The user simply associates a node from the system's existing merchandise hierarchy with a particular APE tree definition (which might then produce more than one tree). Only items that are children of the merchandise hierarchy node are allowed into this tree (or trees). This creates a type of filter on the APE hierarchy tree (or trees) that uses this anchor node. However,

anchor nodes limit the user to filtering only by a particular node in the system's existing merchandise hierarchy and can only be used at the tree level.

The user must assign each APE hierarchy tree set definition an anchor node. Requiring that each tree set have an anchor node ensures that no item appears in more than one tree. Anchor nodes are created by defining keys, but rather are a required property of the tree set definition itself.

Merchandise Mapping xml Configuration File

An XML-based configuration file (found in *<InstallationDirectory>*/modules/pce /etc/APE_Hierarchy.xsd) is used to configure the APE hierarchy.

xml Tag Definitions

The following definitions are used in the xml configuration file.

·**<APE_hierarchy>** – The <APE_hierarchy> begins an individual APE hierarchy. The set can contain multiple hierarchy trees, depending on its configuration.

·**<treeset>** – A <treeset> tag defines the beginning of a particular hierarchy tree configuration

Property	Required	Description
name	No	The name for the <treeset>
mh_node_id	Yes	The anchor node ID, a merchandise hierarchy client load ID
mh_node_level_desc	Yes	The anchor node level description, a merchandise hierarchy client level description. Required because the merchandise hierarchy client load node IDs are not unique across levels.

·**<level>** – The <level> tag begins a level definition within a <treeset> or another <level>. The level defined directly under a <treeset> is the 0th, or root, level.

Property	Required	Description
position	Yes	The integer representing the position of the level within the hierarchy. Root level is 0, first level is 1, etc.

·**<key>** – A <key> node provides filtering for a particular tree, or for a level within a tree, based on a merchandise attribute column and the value of that column.

Property	Required	Description
column	Yes	The merchandise attribute field name.
value	Yes	The value of the merchandise attribute field that <treeset> or <level> applies to.

Example xml file

```
<APE_hierarchy>
<!-- Typical tree set definition -->
  <treeset name="Shoe Tree" mh_node_id="12368"
  mh_node_level_desc="CLASS">
      <level>
            <attr column=CATGORY/>
            <level>
                  <attr column=BRAND/>
                  <level>
                        <attr column=SIZE/>
                  </level>
            </level>
      </level>
    </treeset>
```

The Results of Merchandise Mapping

The results of the merchandise mapping are a set of database tables that describe the APE hierarchy and map client item IDs to their respective APE hierarchy nodes.

APE Hierarchy Node Naming

Each APE hierarchy node should be given a unique name and description that has meaning to the analyst and the client. During the merchandise configuration process, node names are generated using a combination of the tree set name and the column names and values used to generate the node.

Error Checking

Items cannot appear in more than one APE hierarchy tree.

Performing Affinity Analysis

To perform affinity analysis, do the following:

1. Build an affinity tree definition in XML. See "Merchandise Mapping xml Configuration File" on page 7-4 for details.
2. Load the affinity tree. To do this, load the xml configuration file using the commands below. The first command, **apbuilder_run.sh**, loads the xml configuration file into the database. The second command, **pl_create_ape_hierarchy.sh**, creates the hierarchy based on the configuration information provided.

   ```
   > apebuilder.sh -config ape_configuration.xml
   > cd
   ${installdir}/modules/Database/WALNUTSchema/install/oracle/WALNUTSchema/scripts
   > pl_create_ape_hierarchy.sh ${installdir}/config promote_plexports.sh
   ```

3. Once the APE configuration is built and processed, summary sales data must be built which matches the loaded APE configuration tree. This prepares the data for analysis.

   ```
   > cd
   ${installdir}/modules/Database/WALNUTSchema/install/oracle/WALNUTSchema/scripts
   > pl_create_ape_summary.sh ${installdir}/config promote_plexports.sh
   ```

4. Use the APE to perform the affinity parameter estimation. It is assumed you have Perl installed on your system.

   To run the APE, do the following:

   1. Download latest software called "R", which is freely available, from www.r-project.org and follow the installation instructions for your platform.
   2. Install the ROracle package required by APE:

      > R CMD INSTALL ROracle

   3. The Promote distribution includes the R package aper-1.0.tar.gz, the script mainAPE.R, and the configuration file options.R. These files can be found in <*InstallationDirectory*>/modules/pce/ape.

      To install the APE:

      >R CMD INSTALL aper

   4. The file options.R contains configurable APE settings. You must edit the file to define DB login parameters:

      DB.HANDLE='USER/PASSWORD@PROMOTE_INSTANCE'

      The other options can be left at their default settings. The comments in options.R describe the default settings.

   5. To run APE:

      >R CMD BATCH mainAPE.R ape.log

5. Once APE analysis is complete, the results need to be loaded back into the application database. There should be two files, one named bee_ape_price_elasticity.txt and the other named bee_ape_promo_elasticity.txt.

```
> cd
${installdir}/modules/Database/BEECHSchema/install/oracle/BEECHSchema/scripts
> pl_stage_promote.sh ${installdir}/config promote_plexports.sh
${installdir}/modules/Database/BEECHSchema/install/oracle/BEECHSchema/controlfi
les ${deploy}/bee_ape_price_elasticity.txt ${deploy}/bee_ape_promo_
elasticity.txt
> pl_load_ape.sh ${installdir}/config promote_plexports.sh
```

8

Export API

This chapter contains the following:

Introduction

The Export API provides a means for the external access to Promote data. The output data about a promotion is provided for a specified time period either as a text file or an xml file. The authentication process used by Promote is enforced by the API.

Export API

The Export API is available at the server level or through a script-accessible pull client. The client should be started in a shell that can be called by a scheduler or from the command line. The API consists of the following parameters:

Table 8–1 Export API Parameters

Parameter Name: long form (short form)	Default Value	Description	Required?
--server (-s)		The connection URL. If any part of the URL is omitted, the value will be read from the config file if it is present.	No. The default value will be used. If a config file is present, the value will be read from that file.
--user (-u)		The Promote user name. Note that the Promote user action PROMO_EXPORT_PROMO provides access to the Export API. This action is included in the PROMO_AD_PLANNER role.	Yes
--passwd (-p)		The Promote password.	Yes
--command (-c)	list	The method name. For details, see Table 8–2, " Method Descriptions".	No. The default value will be used.

Table 8–1 (Cont.) Export API Parameters

Parameter Name: long form (short form)	Default Value	Description	Required?
--args		Arguments for the method. ■ The dates for the data: From a specific date until a specific date. The final date is either the sysdate or another indicated date. ■ The phase: If a particular phase name is specified and that name contains spaces, then those spaces must be replaced by a "+". If the value ALL is used, all records, filtered by date, are returned. The argument string must be enclosed in double quotes. Arguments must be separated by a semicolon. For example, "01/01/2007_01:01:01;ALL"	No. An argument is required.
--datemask (-d)	MM/DD/YYYY_ HH:MM:SS	The dates that apply to the parameters. The format does not permit any spaces.	No. The default value will be used.
--verbose (-v)	false	The outline steps taken into the standard error.	No. The default value will be used.
--format (-f)	xml	The format of the output. The value is either xml or txt.	No. The default value will be used.
--output (-o)	stdout	Where the output data is directed - either a file or the console (standard output). If a file, the full or relative path is required.	No. The default value will be used.
--config (-cfg)	../conf/promo-pull client.properties	The following properties are included. These properties are used if the --server option does not provide a URL. ■ promote.exporter.servlet.contextroot=promote ■ promote.exporter.servlet.appname=export.do ■ promote.exporter.protocol=http ■ promote.exporter.host=localhost ■ promote.exporter.port=7777 ■ promote.pullclient.datemask=MM/dd/yyyy_HH:mm:ss ■ promote.pullclient.format=xml ■ promote.pullclient.command=list ■ promote.pullclient.timeout=10	No. If not specified, it defaults to the configuration file.
--sysdate (-sd)	The current date and time. Formatted as MM/DD/YYYY_ HH:MM:SS.	For time-sensitive exports, the data up to this date will be returned.	No. The default value will be used.

The following table provides details about the command and argument parameters. Method descriptions can be found in promoteResources.properties, export.help.promotionofferitemsummary, and export.help.promotionofferpositionsummary.

Table 8–2 Method Descriptions

Method Name	Method Description
getPromoOfferItems	The getPromoOfferItems method returns all offers (including information about offer items such as forecast and location) that have changed since time **T** inclusive for active promotions changed since that time for the promotion phase **P** specified. If no existing promotion is matched, then no data is returned. If phase = ALL, then promotions of any phase are returned, including those that have an empty phase field. The item-location metrics (which will be null) are returned even if there is no forecast but the promotion is associated with a store subset.
getPromoOfferPositions	The getPromoOfferPos method returns all offers (including information about the offer position) that have changed since time **T** inclusive for active promotions changed since that time for the promotion phase **P** specified. If no existing promotion is matched, then no data is returned. If phase = ALL, then promotions of any phase are returned, including those that have an empty phase field.
list	The list method returns a list of all available methods (the code and the description) and any pre-defined constants in parentheses). The output will indicate if the parameter is optional.

Method Output

Output is formatted as either text or xml. The fields in the text output are pipe-delimited. The xml output is enclosed in the tags <PromotionOfferItemSummarys> and <PromotionOfferPositionSummarys>. The root tag is PromoteExport. Note that the pre-defined XSLT style sheets provided with the product are configurable.

The promoOfferItems method produces the following output (shown as text output):

PromoExternalName | PromoName | PromoBudget | PromBeginDate | PromoEndDate | PromoLastModified | PromoPhaseName | VehicleExternalName | VehicleName | LocationClientLoadID | LocationLevelDesc | MerchandiseClientLoadID | MerchandiseLevelDesc | OfferExternalName | OfferName | PromoOfferExternalName | PromoOfferName | PromoOfferAmount | OfferTypeExternalName | PromoOfferLastModified | Cost | FullPrice | PromoPrice | ForBaseSales | ForBaseUnits | ForBaseMargin | ForIncSales | ForIncUnits | ForIncMargin | ForAffBaseSales | ForAffBaseUnits | ForAffBaseMargin | ForAffIncSales | ForAffIncUnits | ForAffIncMargin

The promoOfferPos method produces the following output (shown as text output):

PromoExternalName | PromoName | PromBeginDate | PromoEndDate | PromoLastModified | PromoPhaseName | VehicleExternalName | VehicleName | OfferExternalName | OfferName | PromoOfferExternalName | PromoOfferName | PromoOfferAmount | OfferTypeExternalName | PromoOfferLastModified | VehiclePageNumber | PageNum | PageExternalName | XDim | YDim | UnitExternalName | VehiclePagePosName | XBegin | YBegin | XEnd | YEnd | PosNum | Props | Heading | Copy | Coupon | Comments | ImageName | ImageFileName | PositionAmount | PositionTypeExternalName

Examples

Here are some examples, using a pull client, promo-pullclient.sh.

To return all promotions changed since 1 November 2006 in text format:

promo-pullclient.sh --server http://qa-app-l14:7780/promote/export.do **--user** anns **--passwd** anns **--command** getPromoOfferItems **--args** "11/01/2006" **--format** txt

To return a list of available methods:

promo-pullclient.sh -s qa-app-l14:7780 **-u** anns **-p** anns **-c** list **-v**

To display usage information:

promo-pullclient.sh

Export Views

These views should be used to retrieve data in batch mode. They are normally wrapped in the IR views that are customized for a given implementation.

Table 8–3 Export Views

View	Description
PR_PROMO_FULL_SUM_VW	List of non-historical promotions.
PR_PROMO_VERS_FULL_SUM_VW	List of non-historical promotions, along with versions (location or storeset/subset).
PR_PROMO_OFFER_FULL_SUM_VW	List of promotion offers.
PR_PROMO_OFR_VERS_FULL_SUM_VW	List of promotion offers, along with versions (location or storeset/subset).
PR_PROMO_OFFER_CAT_FULL_SUM_VW	List of promotion offer categories.
PR_PROMO_OFR_ITEM_FULL_SUM_VW	List of promotion offer items.
PR_PROMO_OFR_STORE_FULL_SUM_VW	List of promotion offer stores (as taken from promotion offer locations and/or storesets/subsets). This view returns locations expanded to the store level.
PR_PROMO_SUBSET_FULL_SUM_VW	List of promotion storesets/subsets.

9

Tasks and Techniques

This chapter contains the following:

Introduction

The commands described in Chapter 10, "Technical Reference" can be used to perform a variety of tasks. The following techniques are described in detail in this chapter:

- Viewing datasets
- Trimming datasets

Viewing a Dataset

Each dataset corresponds to a directory in a file system. Each directory contains subdirectories. Each subdirectory contains binary files that contain the contents of the dataset. This storage design is ideal for a dataflow engine like the PCE. However, in order to view the contents of the analytical datasets, you must do the following:

Using a Text File

Datasets can be exported to a text file and viewed using any text viewer. To export the file, use the kde.sh export command.

Example: kde.sh export -in tae.ds -out tae.txt

This command produces two files. The tae.txt file contains a pipe-delimited flat file containing the data. The tae.txt.schema file lists the columns of the text file and their data types.

Using the Database

Datasets can be sent to the database for viewing. Use the kde.sh dbwrite utility to do this. For more information on the dbwrite command, see Appendix 10, "Technical Reference".

Processing Incremental Data

This section contains a detailed process for staging and loading incremental data and using Promote Intelligence tools to analyze the data.

1. Stage and load a month of promotion history:

   ```
   bash$ cd ${installdir}/modules/Database/BEECHSchema/install/oracle/BEECH
   Schema/scripts
   bash$ bash pl_stage_promote.sh ${installdir}/config promote_plexports.sh
   ${installdir}/modules/Database/BEECHSchema/install/oracle/BEECHSchema/
   controlfiles bee_promotions.txt
   bash$ bash pl_load_promote.sh ${installdir}/config promote_plexports.sh
   ```

2. Stage a month of transaction data:

   ```
   bash$ pl_stage_promote.sh ${installdir}/config promote_plexports.sh
   ${installdir}/modules/Database/BEECHSchema/install/oracle/BEECHSchema/
   controlfiles bee_mb_detail.txt
   bash$ bash pl_loadP_promote.sh ${installdir}/config promote_plexports.sh
   ```

3. Delete the existing temporary dataset:

   ```
   bash$ delete.sh -y temp.baseline.ds
   ```

4. Run baseline.sh on the new data and store the results in a temporary dataset (to facilitate making changes if necessary):

   ```
   base$ baseline.sh -out temp.baseline.ds -startdate 2006-12-01 -enddate 2007-02-07
   INFO: baseline Run ID: 19 Run Date 2007-02-08 12:30:00
   ```

5. Check mismatch step.

6. Run tae.sh on three new ad events and store the results in a temporary dataset (to facilitate making changes to the tae analysis if necessary). Note that the input data is from the temporary dataset from Step 4:

   ```
   bash$ delete.sh -y temp.tae.ds
   bash$ tae.sh -inhistory temp.baseline.ds -adeventid ROP-7789
   INFO: tae Run ID: 20 Run Date: 2007-02-08 12:30:00
   bash$ tae.sh -inhistory temp.baseline.ds -adeventid ROP-7790  -append
   INFO: tae Run ID: 21 Run Date: 2007-02-08 12:30:00
   bash$ tae.sh -inhistory temp.baseline.ds -adeventid ROP-7791  -append
   INFO: tae Run ID: 22 Run Date: 2007-02-08 12:30:00
   ```

7. Step 6 can be repeated after changing some TAE parameters, again using the temporary dataset:

   ```
   bash$ delete.sh -y temp.tae.ds
   bash$ tae.sh -inhistory temp.baseline.ds -adeventid ROP-7789 -append
   INFO: tae Run ID: 23 Run Date: 2007-02-08 12:30:00
   bash$ tae.sh -inhistory temp.baseline.ds -adeventid ROP-7790  -append
   INFO: tae Run ID: 24 Run Date: 2007-02-08 12:30:00
   bash$ tae.sh -inhistory temp.baseline.ds -adeventid ROP-7791 -append
   INFO: tae Run ID: 25 Run Date: 2007-02-08 12:30:00
   ```

8. Copy the temporary dataset to the database for detailed analysis. A report can be used to examine the results:

   ```
   bash$ tae_temp_stage.sh -in temp.tae.ds
   bash$ tae_temp_load.sh
   ```

9. After an examination of the data, the first run is judged better than the second. Clean the temporary dataset, append the actual dataset, and then load it into the operational TAE dataset:

   ```
   bash$ prune.sh -set temp.tae.ds -criteria run_id>22
   bash$ append.sh -in temp.tae.ds -out tae.ds
   bash$ delete.sh -y temp.tae.ds
   bash$ tae_load.sh
   ```

10. Copy the baseline data to the baseline dataset and then delete the temporary dataset:

    ```
    bash$ append.sh -in temp.baseline.ds -out baseline.ds
    bash$ delete.sh -y temp.baseline.ds
    ```

11. Run market basket analytics. This process can be repeated, if necessary:

    ```
    bash$ pl_create_summaries.sh -start 2007-01-01 -end 2007-01-08
    ```

12. Rebuild the RDM. This process can be repeated, if necessary, as the command re-builds the RDM completely:

    ```
    bash$ pl_load_promote_rdm.sh
    ```

10

Technical Reference

This chapter contains the following:

Introduction

This chapter contains useful reference material, including configuration file details, information about database scripts, and command options and attributes for kde.sh.

Configuration Files

This section contains information about kde.properties and kde-local.vars.

kde.properties

The following is a subset of the properties included in kde.properties. These properties are configured during the installation procedure.

Table 10–1 kde.properties

Property	Description
com.netperceptions.kde.tempDirPath=%{KDE_TMP_DIR_PATH}%	The temp space allocated to the PCE. It should be close to the size of the dataset storage.
com.netperceptions.kde.numberOfCPUs=%{KDE_NUM_CPUS}%	The number of CPUs that the host machine that has the PCE should optimize to. Dual core and hyper-treading CPUs should count as 2 each.
com.netperceptions.kde.rmi.server.port=%{KDE_RMI_SERVER_PORT}%	The port for the PCE Prediction Engine to run. The default port is 11269.
com.netperceptions.kde.rmi.factory.server.port=%{KDE_RMI_FACTORY_SERVER_PORT}%	The default server port is 0 (i.e., any available port).

Table 10–1 (Cont.) kde.properties

Property	Description
com.netperceptions.kde.rmi.predictor.server.port=%{KDE_RMI_PREDICTOR_SERVER_PORT}%	The default server port is 0 (i.e., any available port).
com.netperceptions.kde.rmi.server.ServerDebug=false	Specifies whether the Prediction Engine runs in debug mode. In debug mode, the temporary files for a request are not deleted afterwards.
com.netperceptions.kde.rmi.server.ModelApplyDebug=false	Specifies whether AdModelApply is executed in debug mode by the Prediction Engine.
com.netperceptions.kde.rmi.server.PromoDataValidation=false	Specifies whether the Prediction Engine should do additional validation of the data received.
com.netperceptions.kde.rmi.server.admodelapply.execute.external=false	Specifies how to run applications in a separate jvm or local to a pce server.
com.netperceptions.kde.rmi.server.affinitypredictor.execute.external=false	Specifies how to run applications in a separate jvm or local to a pce server.
com.netperceptions.kde.rmi.server.RunAffinity=true	Specifies whether or not the pce server runs the affinity prediction.
com.netperceptions.kde.rmi.server.Affinity.UseFarTargets=true	Specifies whether or not the affinity prediction includes the far target nodes.
com.netperceptions.kde.rmi.server.ForecastRefactorFlag=true	If true, new code will execute.
com.netperceptions.kde.rmi.server.ReturnAllPromotionItemFlag=false	This flag ensures that all the item predictions are returned to the clients. For example, if the PCE is not able to get product baseline for an item, then the item will be returned with values set to null but with a status of "no predict baseline found." This flag is valid only if the ForecastRefactorFlag is true.
com.netperceptions.kde.rmi.server.DumpAllPromotionItemsFlag=false	This flag updates the log with all items. This flag is valid only if the ForecastRefactorFlag is true.
com.netperceptions.kde.rmi.server.EnableStopWatchFlag=false	This flag ensures that each process is timed with a stopwatch. This flag is valid only if the ForecastRefactorFlag is true.
com.netperceptions.kde.rmi.server.archivePCEForecastStatusResultFlag=false	This flag dumps all the results generated by the PCE into a preconfigured table in the database, using a batch process. This flag is valid only if the ForecastRefactorFlag is true.
com.netperceptions.kde.rmi.server.RGIndicatorFlag=true	This flag enables or disables the Forecast Accuracy Indicator in the PCE.

kde-local.vars

The /etc/kde-local.vars file contains environment variables used by promote shell scripts. It includes the following settings:

- DEBUG_MODE – Determines if the steps in model_build.sh are printed to the console. If this is set to anything, then it is active.
- DWH_HOST – The host name of the database server used by the system.
- DWH_INSTANCE – The instance name of the database server used by the system.
- DWH_ALIAS – The alias of the database server used by the system.
- DWH_USERNAME – The username for the database used by the system.
- DWH_PASSWORD – The password for the database used by the system.
- DWH_TYPE – The database type used by the system. Must be set to *oracle*.
- DWH_PORT – The port number of the database server used by the system.
- TEMP_DIR – The temporary directory used by model_build.sh.
- CP_INITIAL_LIMIT – The initial number of connections that the pool should have when created.
- CP_MIN_LIMIT – The minimum number of connections in the pool. A value of 0 indicates that the connections have been created as required.
- CP_MAX_LIMIT – The maximum number of connections in the connection pool. A value of 0 indicates that there is no maximum limit.
- LOCATION_SUMMARY_LEVEL –The level in the location hierarchy at which analysis is done by the system. This setting is client-specific.
- MERCHANDISE_SUMMARY_LEVEL – The level in the merchandise hierarchy at which analysis is done by the system. This setting is client-specific.

Note: In order to run PCE scripts, you must add <installdir>/modules/pce/bin to the PATH environment variable.

If it is necessary to change any of the settings in this file, it is recommended that you configure kde-local.vars, shown in the following code example.

```
#!/bin/bash
#uncomment DEBUG_MODE to get more messages
#DEBUG_MODE=true

#
# Database settings
#
DWH_HOST=%{database.commondb.oracle.address}%
DWH_INSTANCE=%{database.commondb.oracle.dbname}%
DWH_ALIAS=%{database.commondb.oracle.dbalias}%
DWH_USERNAME=%{database.commondb.oracle.auth.commonoracleauth.user}%
DWH_PASSWORD=%{database.commondb.oracle.auth.commonoracleauth.password}%
DWH_URL=%{database.commondb.oracle.dburl}%
DWH_TYPE=oracle
DWH_PORT=%{database.commondb.oracle.dbport}%

TEMP_DIR=%{KDE_TMP_DIR_PATH}%

# Connection Pool Parameters
CP_INITIAL_LIMIT=5
CP_MIN_LIMIT=5
```

```
CP_MAX_LIMIT=20

# SUMMARY LEVELS
LOCATION_SUMMARY_LEVEL=7
MERCHANDISE_SUMMARY_LEVEL=4

# set global options (e.g., useClearancePeriods) to predictbaseline application
#PBL_OPTIONS="-useClearancePeriod -useDarkPeriods"

#Custom Column Name ros_lift_mean=pbl.ROS_LIFT_MEAN or any other column, only
numeric values will work
#MODEL_APPLY_CUSTOM_COLUMN_NAME="xxx=pbl.xxx,yyy=prod.YYY"
```

Database Scripts

This section contains details about database scripts.

General Loading and Staging Scripts

The following scripts are used to load specific data files into the database and to transfer data from staging tables to target tables. These scripts must be run from the ${installdir}/modules/Database/ASHSchema/install/oracle/ASHSchema/scripts directory.

pl_stage_file.sh

Usage: pl_stage_file.sh [OPTION] ... [FILE] ...

Description: This script runs the specified load procedure to transfer data from the staging tables to the target tables.

Options:

Table 10–2 pl_stage_file.sh Options

-a DIR	--logdir_archive=DIR	directory to archive old log files
-c DIR	--controldir=DIR	directory with data control files
-e NUM	--errorthreshold=NUM	number of errors to allow in load (for DB2, it is a warning threshold)
-l DIR	--logdir=DIR	directory to store logs
-r DIR	--configroot=DIR	configuration root directory
-f FILENAME	--paramfile=FILENAME	filename of DB parameter file in configroot
-h	--help	displays help and exits

pl_load_data.sh

Usage: pl_load_data.sh [OPTION] ... [FILE] ...

Description: This script loads a specified data file into the database.

Options:

Table 10–3 pl_load_data.sh Options

-a DIR	--logdir_archive=DIR	directory to archive old log files
-e NUM	--errorthreshold=NUM	number of errors to allow in load (overwrites the procedure's default limit)
-l DIR	--logdir=DIR	directory to store logs
-r DIR	--configroot=DIR	configuration root directory
-f FILENAME	--paramfile=FILENAME	filename of DB parameter file in configroot
-rlo	--runloadonly	do not run pre and post step
-ne	--noexit	does not exit with a value
-h	--help	displays help and exits

pl_exec_stored_procedure.sh

Usage: pl_exec_stored_procedure.sh [OPTION] ... [FILE] ...

Description: This script runs the procedures in the database.

Options:

Table 10–4 pl_exec_stored_procedure.sh Options

-a DIR	--logdir_archive=DIR	directory to archive old log files
-l DIR	--logdir=DIR	directory to store logs
-r DIR	--configroot=DIR	configuration root directory
-f FILENAME	--paramfile=FILENAME	filename of DB parameter file in configroot
-h	--help	displays help and exits

RDM Scripts

The following scripts are used for RDM maintenance. They should be run from the ${installdir}/modules/Database/SEQUOIASchema/install/oracle/SEQUOIASchema/scripts directory.

pl_create_rdm_summary.sh

Usage: pl_create_rdm_summary.sh <configRoot> <paramFile>

Description: This script creates the RDM views at the aggregation levels specified in the configuration parameters.

pl_load_promote_rdm.sh

Usage: pl_load_promote_rdm.sh <configRoot> <paramFile>

Description: This script loads the base data (promotions, calendar, locations, and merchandise) into the RDM schema.

pl_refresh_arm_sets.sh

Usage: pl_refresh_arm_sets.sh <configRoot> <paramFile>

Description: This script refreshes the data in the RDM schema for ARM sets.

Database Maintenance and Database Functional Scripts

The following scripts are used for functional and maintenance tasks related to the database. They must be run from the ${installdir}/modules/Database/WALNUTSchema/install/oracle/WALNUTSchema/scripts directory.

pl_create_ape.sh

Usage: pl_create_ape.sh <configRoot> <paramFile> <tree_set_id> <create_option>

Description: This wrapper shell script creates the ape hierarchy and the ape summary.

pl_create_map_views.sh

Usage: pl_create_map_views.sh <configRoot> <paramFile>

Description: This shell script calls the procedure that creates pr_merchandise_map_view.

pl_create_merch_loc_seas_elast.sh

Usage: pl_create_merch_loc_seas_elast.sh <configRoot> <paramFile>

Description: This script creates a normalized view for seasonality and price elasticity for the item/location combinations that appear in the transaction data.

pl_create_summaries.sh

Usage: pl_create_summaries.sh <configRoot> <paramFile> <YYYY-MM-DD> <YYYY-MM-DD>

Description: This script aggregates sales transaction data. The summaries are created at different levels, based on the configuration parameters. Different objects are populated by this process.

pl_create_summary_tables.sh

Usage: pl_create_summary_tables.sh <configRoot> <paramFile>

Description: This script creates the physical market basket summary tables in the database. The tables are created at different levels, based on the configuration parameters.

pl_exec_stored_procedure.sh

Usage: pl_exec_stored_procedure.sh <configRoot> <paramFile> <logdir> <loadlist>

Description: This script is a helper script that is used to invoke Oracle database procedures directly.

pl_load_promolocation.sh

Usage: pl_load_promolocation.sh <configRoot> <paramFile>

Description: This script creates a snapshot of the existing promotion/location combinations. A table is populated with the cross-reference data.

pl_load_promoteitem.sh

Usage: pl_load_promoitem.sh <configRoot> <paramFile>

Description: This script creates a snapshot of the existing promotion/item combinations. A table is populated with the cross-reference data.

Database Loading and Staging Scripts

The following wrapper scripts contain the corresponding java classes and stored procedures that perform each task. During the process of staging and loading the data to the target tables, the data is validated, based on integrity rules, and, in general, translated in order to convert external IDS into internal Promote IDs. All of these scripts must be run from the ${installdir}/modules/Database/BEECHSchema/install/oracle /BEECHSchema/scripts directory.

pl_load_future_price_cost.sh

Usage: pl_load_future_price_cost.sh <configRoot> <paramFile>

Description: This script merges the future price and cost data.

pl_load_ape.sh

Usage: pl_load_ape.sh <configRoot> <paramFile>

Description: This script merges the APE results for price and promo elasticity.

pl_load_base.sh

Usage: pl_load_base.sh <configRoot> <paramFile>

Description: This script moves base promotion data (calendar, merchandise, location, and hierarchy definitions) from the staging tables into the active tables.

pl_load_like_location.sh

Usage: pl_load_like_location.sh <configRoot> <paramFile>

Description: This script merges like location information.

pl_load_like_merchandise.sh

Usage: pl_load_like_merchandise.sh <configRoot> <paramFile>

Description: This script merges like merchandise information.

pl_load_mb_detail.sh

Usage: pl_load_mb_detail.sh <configRoot> <paramFile>

Description: This script merges the incoming sales transaction data.

pl_load_model_accuracy.sh

Usage: pl_load_model_accuracy.sh< configRoot> <paramFile>

Description: This script moves model accuracy metrics data.

pl_load_parameter.sh

Usage: pl_load_parameter.sh <configRoot> <paramFile>

Description: This script moves seasonality parameter values.

pl_load_predictable.sh

Usage: pl_load_predictable.sh <configRoot> <paramFile>

Description: This script moves calculated predictable data.

pl_load_promo_config.sh

Usage: pl_load_promo_config.sh <configRoot> <paramFile>

Description: This script loads basic promotion configuration data.

pl_load_promo_history.sh

Usage: pl_load_promo_history.sh <configRoot> <paramFile>

Description: This script loads historical promotion data (all promotion elements).

pl_load_seasonality.sh

Usage: pl_load_seasonality.sh <configRoot> <paramFile>

Description: This script loads seasonality maps data.

pl_load_tae.sh

Usage: pl_load_tae.sh <configRoot> <paramFile>

Description: This script loads calculated TAE metrics.

pl_load_tae_temp.sh

Usage: pl_load_tae_temp.sh <configRoot> <paramFile>

Description: This script loads calculated TAE metrics into a temporary location.

pl_load_weekly_history.sh

Usage: pl_load_weekly_history.sh <configRoot> <paramFile> <controlDir> <dataFiles>

Description: This script loads inventory data into the application tables.

pl_migrate_merch_attrs.sh

Usage: pl_migrate_merch_attrs.sh <configRoot> <paramFile>

Description: This script updates the merchandise hierarchy table with the incoming cost, price, and prod_type.

pl_stage_promote.sh

Usage: pl_stage_promote.sh <configRoot> <paramFile> <controlDir> <dataFiles>

Description: This script loads the data from the text data files into the staging tables.

PCE Scripts

The following scripts, listed alphabetically, are part of the Promote toolset. Note that the -kdehelp option provides the full list of command line options.

Note: In order to run PCE scripts, you must add <installdir>/modules/pce/bin to the PATH environment variable.

arm_pull.sh

Usage: arm_pull.sh -minconfidence <conf> -startdate <yyyy-MM-dd> -endate <yyyy-MM-dd> [-kdehelp] [-help]

arm_pull_load.sh

Usage: arm_pull_load.sh [-help]

apebuilder.sh

Usage: apebuilder.sh -config <config_file> [-kdehelp] [-help]

Description: This command loads the ape configuration file into the database. Once it is loaded, the pl_create_ape.sh script can be used to create the ape hierarchy.

append.sh

Usage: append.sh -in <name or location> -out <name or location> [-kdehelp] [-help]

Description: This script is used to append data from a source dataset to a target dataset. This script calls the KDE application copy and uses its -append option. This script allows the user to append a temporary dataset to the main dataset.

arm.sh

Usage: arm.sh -minconfidence <conf> -maxsetsize <size> -runid <id> [-kdehelp] [-help]

Description: This script calls the ARM application. The ARM application is responsible for set mining, association rule generation, and computing a variety of associated metrics used by a retail analyst.

baseline.sh

Usage: baseline.sh -out <dataset name or location> -startdate <yyyy-mm-date> -enddate <yyyy-mm-dd> [-kdehelp] [-help]

Description: In order to add additional baseline data without overwriting the old baseline data, the user must run baseline.sh with the -out option set to a temporary dataset. The user must then append this dataset to the main baseline dataset using append.sh.

baseline_coarse_grained.sh

Usage: baseline_coarse_grained.sh -from_date <yyyy-MM-dd> -to_date <yyyy-MM-dd> -promoweight <0> -clearanceweight <0> -darkweight <0> -batch_size <10000> [-global_options] [-local_options] [-help]

build_models.sh

Usage: build_models.sh [category_number] [-help]

capture_models.sh

Usage: capture_models.sh [-src srcModleSet] [-dest destModelDir] [-help]

delete.sh

Usage: delete.sh -set <Dataset name or location> -type <binary/text-delimited> [-[no-y]

Description: This script is used to delete a dataset. The user is prompted to confirm the delete operation unless the -y flag is provided.

import_models.sh

Usage: import_modles.sh [-src srcModelSet] [-dest destModelDir] [-inactive inactiveModelFile] [-clean] [-help]

init_runctl.sh

Usage: init_runctl.sh

Description: This script uses a dataset to store information about an application run, including run_id, start and end date/time, run duration, run status, run description, and run type. A KDE utility. It has commands for initializing one or more run control datasets, adding records to the dataset that are assigned a unique run ID, updating the status of each record in the dataset, and querying a record based on its run ID.

model_build.sh

Usage: model_build.sh <category_id>

Description: This script is used to build a model for the identified category. The category is identified by its internal ID.

model_prepare.sh

Usage: model_prepare.sh

Description: This script is used to prepare the system for modeling categories. It combines the input data, which consists of promotion history, TAE scorecards, and category-level sales, into two datasets, one for model training and the other for model testing. Use the model_config.xml file to configure the model preparation process.

pceserver.sh

Usage: pceserver -start -shutdown -restart -query status -query version -query uptime -query load -query port -reload port -reload props -reload logging -timeout -help

pl_refresh_arm_sets.sh

Usage: pl_refresh_arm_sets.sh <configRoot> <paramFile>

Description: This script is used to load the results of the ARM tool into the database.

predict_baseline.sh

Usage: predict_baseline.sh -periodid <future_period_id> -period_type <future_period_type> -predict_date <yyyy-mm-dd> -baseline_start <yyyy-mm-dd> -baseline_end <yyyy-mm-dd> [-help]

Description: This script computes the predicted baseline of the input dataset.

predict_baseline_coarse_grained.sh

Usage: predict_baseline_coarse_grained.sh -from_date <yyyy-MM-dd> -to_date <yyyy-MM-dd> -use_avg -batch_size <10000> [-global_options] [-local_options] [-help]

predict_baseline_truncate.sh

Usage: predict_baseline_truncate.sh -installdir <install path>

rgmetric.sh

Usage: rgmetric.sh -run_date <yyyy-MM-dd> -view_Name <purchasetype_b> -batch_size <10000> [-global_options] [-local_options]

tae.sh

Usage: tae.sh -adnameid external_id [-algorithm <algorithm>] [-basewindow yyyy-MM-dd,yyyy-MM-dd,pmin] [-kdehelp] [-help]

Description: This script is used to measure the effectiveness of a given ad event.

tae_dbl.sh

Usage: tae_dbl.sh -adnameid <external_id> [-algorithm algorithm] [-basewindow <yyyy-MM-dd, pmin>] [-kdehelp] [-help]

tae_load.sh

Usage: tae_load.sh [-help]

Description: This script is used to load tae data from the staging tables into the operational tables. It can be re-run if the staging tables have been updated with new data. In such a case, the data in the operational tables is compared to the new data using promotion IDs, and, for a given promotion ID, any old data is replaced with new data. This script calls pl_load_tae.sh and pl_alloc_tae.sh.

tae_load_temp.sh

Usage: tae_load_temp.sh [-help]

Description: This script is similar to the one above. It is used to load temporary tae data from staging tables to operational tables. It loads the data from BEE_TAE_TEMP_METRIC into PR_TAE_TEMP_METRIC.

tae_publish_promo.sh

Usage: tae_publish_promo.sh <promoid>

tae_publish_run.sh

Usage: tae_publish_run.sh <runid>

tae_stage.sh

Usage: tae_stage.sh [-in <name or location>] [-help]

Description: This script is used to stage data. This script truncates the tae staging table, BEE_TAE_METRIC, before adding records.

tae_stage_temp.sh

Usage: tae_stage_temp.sh -in <dataset name or location> [-help]

Description: This script is similar to the one above. It is used to load temporary tae data into staging tables. It requires the name of the temporary dataset. It loads the data into BEE_TAE_TEMP_METRIC.

KDE Syntax

This section contains syntax details for kde.sh baseline, kde.sh tae, and kde.sh arm, kdesh dbread, kde.sh dbwrite, and kde.sh export. These commands are the building blocks of the analytical commands described in "PCE Scripts" on page 10-8. In most cases, these commands should not be used directly. However, the baseline, tae, and arm commands all accept kde options as well. Thus, the kde options described in this section can be passed to their respective wrapper commands.

kde.sh baseline

The baseline.sh script calls kde.sh baseline.

Here is an example of kde.sh baseline:

```
kde.sh baseline
    -indb dbtype=oracle,host=dbhost,instance=retaildb,user=myuser,
        password=password,table=MD_DETAIL
    -out type=binary,name=/bigdisk/dataset/baseline.out.ds
    -genxml baseline.xml
    -backtrace
    -startdate 2002-12-29
    -enddate 2003-03-29
    -append
```

The following table lists the complete set of options for the command.

Table 10–5 kde.sh baseline Command Options

Option	Option Description	Required/ Optional?
-out <out DS attributes>	Baseline output dataset. Takes attributes.	Required
-startdate *yyyy-MM-dd*	Set start of date range for baseline.	Required
-ac_level *column-name*	Name of column to use for ac_level.	Optional
-ad_ind *column-name*	Name of column to use for ad_ind.	Optional
-[no] append	Append output to dataset.	Optional
-[no] backtrace	Display detailed stack backtrace.	Optional
-[no] debug	Turn on debugging.	Optional
-enddate *yyyy-MM-dd*	Set end of date range for baseline.	Optional
-ext_margin_amt *column-name*	name of column to use for ext_margin_amt.	Optional
-ext_retail_amt *column-name*	Name of column to use for ext_retail_amt.	Optional
-genxml *file-name*	Name of file to write pre-processed generated xml to.	Optional
-[no] help	Display message.	Optional
-in <in DS attributes>	Detail input dataset. Takes attributes.	Optional
-inLocMap <in DS attributes>	Location Map input dataset. Takes attributes.	Optional
-inMerchMap <in DS attributes>	Merchandise Map input dataset. Takes attributes.	Optional
-indb <in JDBC connection attributes>	detail input JDBC connection. Takes attributes.	Optional
-loc_client_load_id *column-name*	Name of column to use for loc_client_id.	Optional
-margin	Enable margin metric calculations.	Optional
-merch_client_load_id *column-name*	Name of column to use for merch_client_load_id.	Optional
-nelements *cnt*	Number of elements to read (-1 to all).	Optional
-options *file-name*	Name of property file containing values for command line options.	Optional
-runid *identifier*	Run identifier.	Optional
-[no] ssd	Input dataset contains sku/store/day aggregated data.	Optional
-txn_date *column-name*	Name of column to use for txn_date.	Optional

Table 10–5 (Cont.) kde.sh baseline Command Options

Option	Option Description	Required/ Optional?
-txn_id *column-name*	Name of column to use for txn_id.	Optional
-txn_total *column-date*	Name of column to use for txn_total.	Optional
-units_sold *column-name*	Name of column to use for units_sold.	Optional

Option Attributes

Some of the options in Table 10–5, " kde.sh baseline Command Options" take attributes.

The option **-out** take the following attributes:

- **type** – the type of output set. Values are *text*, *fixed-text*, and *binary*.
- **schema** – the pathname of the schema file.
- **compression** – the compression method used. Values are *none* and *gzip*.
- **name** – the pathname of the dataset. This is required.
- **timezone** – the timezone used.
- **encoding** – the encoding used.
- **delimiter** – the delimiter used.
- **locale** – the locale used.
- **mode** – the mode used. Values are *append* and *overwrite*.

The options **-in, -inLocMap,** and **inMerchMap** take the following attributes:

- type - the type of output set. Values are *text*, *fixed-text*, and *binary*.
- **schema** – the pathname of the schema file.
- **compression** – the compression method used. Values are none and gzip.
- **name** – the pathname of the dataset. This is required.
- **nElements** – the number of elements to read. The default value is *all*.
- **logConvErrors** – indicates if the conversion error should be logged.
- **maxRejects** – the maximum number of parse errors allowed before the application stops.
- **rejectFile** – the pathname of the reject file.

The options **-outdbLocMap** and **-outdbMerchMap** take the following attributes:

- **dbtype** – the type of database to access.
- **host** – the host name of the database to access.
- **port** – the network port number of the database to access.
- **instance** – the instance name of the database to access.
- **user** – the database user name.
- **password** – the database user password.
- **table** – the name of the table to access.

The option **-indb** takes the following attributes:

- **dbtype** – the type of database to access.
- **host** – the host name of the database to access.
- **port** – the network port number of the database to access.
- **instance** – the instance name of the database to access.
- **user** – the database user name.
- **password** – the database user password.
- **table** – the name of the table to access.
- **query** – the query to run instead of accessing the table.

kde.sh tae

The tae.sh script calls kde.sh tae.

Here is an example of kde.sh tae:

```
kde.sh tae
    -indbdetail dbtype=oracle,host=dbhost,instance=retaildb,user=myuser,
        password=password,table=MD_DETAIL
    -inhistory type=binary,name=/bigdisk/dataset/baseline.out.ds
    -out type=binary,name=/bigdisk/dataset/tae.out.ds
    -indbAdEvent dbtype=oracle,host=dbhost,instance=retaildb,user=myuser,
        password=password,table=IR_PR_PROMOTIONS_VW
    -insub type=text-delimited,name=/bigdisk/dataset/item.tae.sub.ds
    -tmpdir /bigdisk/dataset/temp
    -blalloc
    -affinity
    -quality
    -append
    -genxml tae.xml
    -backtrace
    -adeventid 77890
    -algorithm cafe
    -inFocusItems
dbtype=oracle,host=dbhost,instance=retaildb,user=myuser,password=password,
         table=PR_FOCUS_ITEMS
```

The following table lists the complete set of options for the command.

Table 10–6 kde.sh tae Command Options

Option	Option Description	Required/ Optional?
-adeventid	Ad event ID.	Required
-inhistory <in DS attributes>	Historical baseline input dataset.	Required
-ac_id ac *analysis level*	Hierarchy ID to use for affinity correlation analysis.	Optional
-ac_level *column-name*	Name of column to use for ac_level.	Optional
-ad_ind *column-name*	Name of column to use for ad_ind.	Optional
-[no] affinity	Perform affinity.	Optional
-algorithm *algorithm*	The algorithm to use to create the window(s).	Optional
-[no] append	Append output to dataset.	Optional
-[no] backtrace	Display detailed stack backtrace.	Optional

Table 10–6 (Cont.) kde.sh tae Command Options

Option	Option Description	Required/ Optional?
-basewindow *yyyy-MM-dd, yyyy-MM-dd, pmin*	The set of baseline windows. Can be repeated.	Optional
-[no] blalloc	Perform baseline allocation.	Optional
-[no] debug	Turn on debugging.	Optional
-ext_margin_amt *column-name*	Name of column to use for ext_margin_amt.	Optional
-ext_retail_amt *column-name*	Name of column to use for ext_retail_amt.	Optional
-genxml *file-name*	Name of file to write pre-processed generated xml to.	Optional
-[no] groupsort	Use group sort.	Optional
-[no] help	Display message.	Optional
-inAdEvent <in DS attributes>	Ad event input dataset. Takes attributes.	Optional
inFocusItems <in DS attributes>	Focus items input dataset. Takes attributes.	Optional
-inLocMap <in DS attributes>	Location Map input dataset. Takes attributes.	Optional
-inMerchMap <in DS attributes>	Merchandise Map input dataset. Takes attributes.	Optional
-indbAdEvent <in JDBC connection attributes>	Ad event input JDBC connection. Takes attributes.	Optional
indbFocusItems <in JDBC connection attributes>	Focus items input JDBC connection. Takes attributes.	Optional
-indbdetail <in JDBC connection attributes>	Detail input JDBC connection. Takes attributes.	Optional
-indetail <in DS attributes>	Detail input dataset. Takes attributes.	Optional
-insub <in DS attributes>	Substitution input dataset. Takes attributes.	Optional
-item_id *analysis level*	Hierarchy ID to use for analysis.	Optional
-loc_client_load_id *column-name*	Name of column to use for loc_client_id.	Optional
-[no] memoryreport	Print memory usage between steps.	Optional
-merch_client_load_id *column-name*	Name of column to use for merch_client_load_id.	Optional
-minconf *conf*	Minimum confidence (0.0).	Optional
-minfreq *freq*	Minimum frequency (0.0).	Optional
-minlift *lift*	Minimum lift (5.0).	Optional
-minwindows *cnt*	Minimum number of windows (1).	Optional
-nelements *cnt*	Number of elements to read (-1 to all).	Optional
-options *file-name*	Name of property file containing values for command line options.	Optional
-out <out DS attributes>	Metric output dataset. Takes attributes.	Optional
-outaffinity <out DS attributes>	Affinity output dataset. Takes attributes.	Optional
-outdb <out JDBC connection attributes>	Metric output JDBC connection. Takes attributes.	Optional
-[no] quality	Perform quality metrics.	Optional
-runid *identifier*	Run identifier.	Optional
-tmpdir *dir*	Directory for temporary files.	Optional

Table 10–6 (Cont.) kde.sh tae Command Options

Option	Option Description	Required/ Optional?
-txn_date *column-name*	Name of column to use for txn_date.	Optional
-txn_id *column-name*	Name of column to use for txn_id.	Optional
-units_sold *column-name*	Name of column to use for units_sold.	Optional

Option Attributes

Some of the options in Table 10–6, " kde.sh tae Command Options" take attributes.

The options **-inhistory, -inAdEvent, -infocusItems, -inLocMap, -inMerchMap, -indetail,** and **-insub** take the following attributes:

- **type** – the type of input set. Values are *text*, *fixed-text*, and *binary*.
- **schema** – the pathname of the schema file.
- **compression** – the compression method used. Values are *none* and *qzip*.
- **name** – the pathname of the dataset. This is required.
- **nElements** – the number of elements to read. The default is *all*.
- **logConvErrors** – indicates if the conversion error should be logged.
- **maxRejects** – the maximum number of parse errors allowed before the application stops.
- **rejectFile** – the pathname of the reject file.

The options **-indbAdEvent, -indbFocusItems,** and **-indbdetails** take the following attributes:

- **dbtype** – the type of database to access.
- **host** – the host name of the database to access.
- **port** – the network port number of the database to access.
- **instance** – the instance name of the database to access.
- **user** – the database user name.
- **password** – the database user password.
- **table** – the name of the table to access.
- **query** – the query to run instead of accessing the table.

The options **-out** and **-outaffinity** take the following attributes:

- **type** – the type of output set. Values are *text*, *fixed-text*, and *binary*.
- **schema** – the pathname of the schema file.
- **compression** – the compression method used. Values are *none* and *gzip*.
- **name** – the pathname of the dataset. This is required.
- **timezone** – the timezone used.
- **encoding** – the encoding used.
- **delimiter** – the delimiter used.
- **locale** – the locale used.

- **mode** – the mode used. Values are *append* and *overwrite*.

The options **-outdb, -outdbLocMap** and **-outdbMerchMap** take the following attributes:

- **dbtype** – the type of database to access.
- **host** – the host name of the database to access.
- **port** – the network port number of the database to access.
- **instance** – the instance name of the database to access.
- **user** – the database user name.
- **password** – the database user password.
- **table** – the name of the table to access.

kde.sh arm

The tae.sh script calls kde.sh arm.

The following table lists the complete set of options for the command.

Table 10–7 kde.sh arm Command Options

Option	Option Description	Required/ Optional?
-outconst <out DS attributes>	Set constants output dataset. Takes attributes.	Required
-outitems <out DS attributes>	Items output dataset. Takes attributes.	Required
-ac_level *column-name*	Affinity correlation level.	Optional
-ad_ind *column-name*	Name of column to use for ad_ind.	Optional
-[no] adindicator	Use of included ad_ind field for items on ad.	Optional
-[no] append	Append output to dataset.	Optional
-[no] backtrace	Display detailed stack backtrace.	Optional
-[no] debug	Turn on debugging.	Optional
-enddate *yyyy-MM-dd*	Set end of date range for transaction.	Optional
-ext_margin_amt *column-name*	Name of column to use for ext_margin_amt.	Optional
-ext_net_margin_amt *column-name*	Name of column to use for ext_net_margin_amt.	Optional
-ext_retail_amt *column-name*	Name of column to use for ext_met_margin.	Optional
-genxml *file-name*	Name of file to write pre-processed generated xml to.	Optional
-[no] gm	Dataset has gross margin field.	Optional
-[no] help	Display message.	Optional
-in <in DS attributes>	Detail input dataset. Takes attributes.	Optional
-inMerchMap <in DS attributes>	Merchandise Map input dataset. Takes attributes.	Optional
-indb <in JDBC connection attributes>	Detail input JDBC connection. Takes attributes.	Optional
-itemdelimiter *delimiter*	Item delimiter for concatenated item_id fields.	Optional
-maxrules *limit*	Maximum number of rules to produce.	Optional
-maxsetsize *size*	.Maximum set size to look for.	Optional
-merch_client_load_id *column-name*	Name of column to use for merch_client_load_id.	Optional

Table 10–7 (Cont.) kde.sh arm Command Options

Option	Option Description	Required/ Optional?
-minconfidence *conf*	Minimum confidence for a rule (0.5).	Optional
-minfrequency *freq*	Minimum frequency for a set (0.0).	Optional
-minrconfidence *rconf*	Minimum reverse confidence for a rule (0.02).	Optional
-minsupport *cnt*	Minimum support for a set (0.0).	Optional
-nelements *cnt*	Number of elements to read (-1 to all).	Optional
-[no] nm	Dataset has net mergin field.	Optional
-options *file-name*	Name of property file containing values for command line options.	Optional
-outrules <out DS attributes>	Rules output dataset. Takes attributes.	Optional
-outruleslhs <out DS attributes>	Rules lhs output dataset. Takes attributes.	Optional
-outrulesrhs <out DS attributes>	Rules rhs output dataset. Takes attributes.	Optional
-outsetsum <out DS attributes>	Set summary output dataset. Takes attributes.	Optional
-partition_id *id*	Operate on supplied partition id (0.0).	Optional
-[no] rules	Produce rules.	Optional
-runid *id*	Prepend each row with supplied id (0,0).	Optional
-[no] setmetrics	Produce set/rule metrics.	Optional
-startdate *yyyy-MM-dd*	Set start of date range for transaction.	Optioanl
-txn_id *column-name*	Name of column to use for txn_id.	Optional
-units_sold *column-name*	Name of column to use for units_sold.	Optional

Option Attributes

Some of the options in Table 10–7, " kde.sh arm Command Options" take attributes.

The options **-in** and **-inMerchMap** take the following attributes:

- **type** – the type of input set. Values are *text*, *fixed-text*, and *binary*.
- **schema** – the pathname of the schema file.
- **compression** – the compression method used. Values are *none* and *qzip*.
- **name** – the pathname of the dataset. This is required.
- **nElements** – the number of elements to read. The default is *all*.
- **logConvErrors** – indicates if the conversion error should be logged.
- **maxRejects** – the maximum number of parse errors allowed before the application stops.
- **rejectFile** – the pathname of the reject file.

The option **-indb** take the following attributes:

- **dbtype** – the type of database to access.
- **host** – the host name of the database to access.
- **port** – the network port number of the database to access.
- **instance** – the instance name of the database to access.

- **user** – the database user name.
- **password** – the database user password.
- **table** – the name of the table to access.
- **query** – the query to run instead of accessing the table.

The options **-outconst, -outitems, -outrules, -outruleslhs, -outrulesrhs,** and **-outsetsum** take the following attributes:

- **type** – the type of output set. Values are *text*, *fixed-text*, and *binary*.
- **schema** – the pathname of the schema file.
- **compression** – the compression method used. Values are *none* and *gzip*.
- **name** – the pathname of the dataset. This is required.
- **timezone** – the timezone used.
- **encoding** – the encoding used.
- **delimiter** – the delimiter used.
- **locale** – the locale used.
- **mode** – the mode used. Values are *append* and *overwrite*.

kde.sh dbread

The kde.sh dbread command is used to read a table from the database to a dataset.

The following table lists the complete set of options for the command.

Table 10–8 kde.sh dbread Command Options

Option	Option Description	Required/ Optional?
-out <out DS attributes>	Output dataset. Takes attributes.	Required
-[no] append	Append output to dataset.	Optional
-[no] backtrace	Display detailed stack backtraces.	Optional
-[no] debug	Turn on debugging.	Optional
-genxml *file-name*	Name of file to write pre-processed, generated xml to.	Optional
-[no] help	Display message.	Optional
-host *host-name*	Host name of database to access.	Optional
-instance *database-instance*	Instance name of database to access.	Optional
-nelements *cnt*	Number of elements to read (-1 to all)	Optional
-options *file-name*	Name of property file containing values for command line options.	Optional
-password *password*	User password.	Optional
-port *port-number*	Network port number of database to access.	Optional
-query *query-string*	Query to run.	Optional
-runid *identifier*	Run identifier.	Optional

Table 10–8 (Cont.) kde.sh dbread Command Options

Option	Option Description	Required/ Optional?
-table *table-name*	Name of table to access.	Optional
-type *database-type*	Type of database to access.	Optional
-user *user-name*	User name.	Optional

Option Attributes

Some of the options in Table 10–8, " kde.sh dbread Command Options" take attributes.

The option **-out** take the following attributes:

- **type** – the type of output set. Values are *text*, *fixed-text*, and *binary*.
- **schema** – the pathname of the schema file.
- **compression** – the compression method used. Values are *none* and *gzip*.
- **name** – the pathname of the dataset. This is required.
- **timezone** – the timezone used.
- **encoding** – the encoding used.
- **delimiter** – the delimiter used.
- **locale** – the locale used.
- **mode** – the mode used. Values are *append* and *overwrite*.

kde.sh dbwrite

The kde.sh dbwrite command is used to write a dataset into a table. The schema of the table and the dataset must match.

The following table lists the complete set of options for the command.

Table 10–9 kde.sh dbwrite Command Options

Option	Option Description	Required/ Optional?
-in <in DS attributes>	Output dataset. Takes attributes.	Required
-table *table-name*	Name of table to access.	Required
-[no] append	Append output to dataset.	Optional
-[no] backtrace	Display detailed stack backtraces.	Optional
-batchsize *positive-integer*	Number of transactions per batch.	Optional
-commitsize *positive-integer*	Number of transactions per commit block.	Optional
-[no] debug	Turn on debugging.	Optional
-genxml *file-name*	Name of file to write pre-processed, generated xml to.	Optional
-[no] help	Display message.	Optional
-host *host-name*	Host name of database to access.	Optional
-instance *database-instance*	Instance name of database to access.	Optional
-nelements *cnt*	Number of elements to read (-1 to all)	Optional

Table 10–9 (Cont.) kde.sh dbwrite Command Options

Option	Option Description	Required/ Optional?
-options *file-name*	Name of property file containing values for command line options.	Optional
-parallel *positive-integer*	Number of parallel instances of database accessor to start.	Optional
-password *password*	User password.	Optional
-port *port-number*	Network port number of database to access.	Optional
-[no] truncate	Whether or not to truncate table before writing to it.	Optional
-type *database-type*	Type of database to access.	Optional
-user *user-name*	User name.	Optional

Option Attributes

Some of the options in Table 10–9, " kde.sh dbwrite Command Options" take attributes.

The options **-in** takes the following attributes:

- **type** – the type of input set. Values are *text*, *fixed-text*, and *binary*.
- **schema** – the pathname of the schema file.
- **compression** – the compression method used. Values are *none* and *qzip*.
- **name** – the pathname of the dataset. This is required.
- **nElements** – the number of elements to read. The default is *all*.
- **logConvErrors** – indicates if the conversion error should be logged.
- **maxRejects** – the maximum number of parse errors allowed before the application stops.
- **rejectFile** – the pathname of the reject file.

kde.sh export

The kde.sh export command is used to export a dataset into a flat file. the data is pipe-delimited.

The following table lists the complete set of options for the command.

Table 10–10 kde.sh export Command Options

Option	Option Description	Required/ Optional?
-in <in DS attributes>	Output dataset. Takes attributes.	Required
-out <out DS attributes>	Output dataset. Takes attributes.	Required
-[no] append	Append output to dataset.	Optional
-[no] backtrace	Display detailed stack backtraces.	Optional
-[no] debug	Turn on debugging.	Optional
-genxml *file-name*	Name of file to write pre-processed, generated xml to.	Optional

Table 10–10 (Cont.) kde.sh export Command Options

Option	Option Description	Required/ Optional?
-[no] help	Display message.	Optional
-nelements *cnt*	Number of elements to read (-1 to all)	Optional
-options *file-name*	Name of property file containing values for command line options.	Optional

Option Attributes

Some of the options in Table 10–9, " kde.sh dbwrite Command Options" take attributes.

The options **-in** takes the following attributes:

- **type** – the type of input set. Values are *text*, *fixed-text*, and *binary*.
- **schema** – the pathname of the schema file.
- **compression** – the compression method used. Values are *none* and *qzip*.
- **name** – the pathname of the dataset. This is required.
- **nElements** – the number of elements to read. The default is *all*.
- **logConvErrors** – indicates if the conversion error should be logged.
- **maxRejects** – the maximum number of parse errors allowed before the application stops.
- **rejectFile** – the pathname of the reject file.

The option **-out** take the following attributes:

- **type** – the type of output set. Values are *text*, *fixed-text*, and *binary*.
- **schema** – the pathname of the schema file.
- **compression** – the compression method used. Values are *none* and *gzip*.
- **name** – the pathname of the dataset. This is required.
- **timezone** – the timezone used.
- **encoding** – the encoding used.
- **delimiter** – the delimiter used.
- **locale** – the locale used.
- **mode** – the mode used. Values are *append* and *overwrite*.

A

Sample Dataset

This appendix describes some of the features of the KSInc dataset distributed with Oracle Retail Promotion Intelligence (Promotion Intelligence) and Oracle Retail Promotion Planning and Optimization (Promotion Planning).

This appendix contains the following sections:

About KSInc

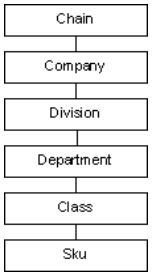
KSInc is a fictional retailer used to illustrate the functional capabilities of the product set. In addition, the dataset can be used as a model for simple customer implementations.

Company Overview

KSInc is a general retailer that sells a variety of merchandise. Most of the sample data distributed with the product is under the Core Toys division.

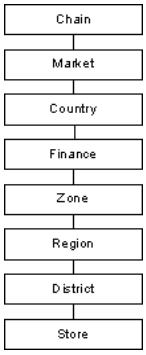
Merchandise Hierarchy

KSInc is deployed with a simple merchandise hierarchy with 6 levels.



Location Hierarchy

The location hierarchy consists of 8 levels.



Organization Structure

This section describes the organization structure.

Roles

The following table illustrates the roles and responsibilities for KSInc.

Table A–1 Roles and Responsibilities

Role Name	Description	Master Data	Calendar	Promotions	Vehicle	Items
PROMO_BUSINESS_ ADMIN	Business Administrator - A business user role that manages the business data setup. Although this role may take on other responsibilities, this role is primarily responsible for tasks such as maintaining data, managing templates, and so on.	Create Edit View				
PROMO_AD_PLANNER	Ad Planner - A marketing person who thinks about the larger promotional calendar. This user should have the ability to edit and create calendar events, create promotions, etc. This is not a person responsible for configuration, defining templates, etc.	View	Create Edit View	Create Edit View Export Reports	Create Edit View	View

Table A–1 (Cont.) Roles and Responsibilities

Role Name	Description	Master Data	Calendar	Promotions	Vehicle	Items
PROMO_MERCH_ PLANNER	For KSInc, merchandise planners are those that are assigned to execute a merchandising plan for a part of KSInc's business. Used in conjunction with the CATEGORY_ MANAGER role below.	View	View	Edit View	View	
PROMO_CATEGORY_ MANAGER	Category Manager - A person who is directly responsible for one or more specific categories of merchandise. Typically used in conjunction with the MERCHANDISE_ PLANNER role. CATEGORY_MANAGER is usually assigned at a given point in the merchandise hierarchy and MERCHANDISE_ PLANNER is the role assigned at the chain level.	View Create Edit				PROMO_ CATEGORY_ MANAGER
PROMO_EXEC	Marketing Executive - An executive who monitors performance across all merchandise. While not responsible for editing or executing the promotional strategy, this user would expect to be able to see what is happening both at a high level and at a low level if desired.	View	View	View Export Reports	View	View

The meaning of each access area is defined below.

- **Master Data** – Master Data refers to foundation data used to configure and maintain the product. Examples include page, vehicle, and promotion templates.
- **Marketing Calendar** – The calendar refers to the time based marketing plan, the Calendar screen within the application and its associated events. For example, for a user to create a competitive event on the shared marketing calendar access has to be granted to the marketing calendar.
- **Promotions** – Promotions are the documents used to plan advertisements. A user must have promotion view rights in order to see these advertising plans.
- **Vehicle** – A vehicle is a specific mechanism or tactic used to execute a promotion, for example a circular. Vehicle rights are required in order to change the layout or allocation of a specific media. For example, to allocate categories to existing whitespace, create additional vehicles, or change the number of pages a user has to have vehicle editing rights.
- **Items** – Items refer to specific SKUs or groups of SKUs on a vehicle. In the context of roles and responsibilities, this refers to rights that grant a user the ability to manage (choose) merchandise from a subset of the merchandise hierarchy. Only those users with Items access can make decisions about specific items to promote.

Users

The following table lists the users and roles configured in the application.

Table A–2 Users and Roles

Full Name	Username/Password	Role
Rodney Tudor	rodneyt	Executive
Steve Calhoon	stevec	Executive
Bernard Allen	bernarda	Business Admin
Geof Rogers	geofr	Ad Planner
Ann Smith	anns	Ad Planner
Vladimir Olson	vladimiro	Ad Planner
Bob Hashimoto	bobh	Ad Planner
Jason Hawthorn	jaysonh	Merch Planner + Cat Mgr Games
Kerry O'Leary	kerryo	Merch Planner + Cat Mgr Construction Toys
Joseph Hunter	josephh	Merch Planner+ Cat Mgr Action Figures
Devin Pritchard	devinp	Merch Planner + Cat Mgr Puzzles
Nick Bosworth	nickb	Merch Planner + Cat Mgr Barbie and Accessories
Stephanie Tauzell	stephaniet	Merch Planner + Cat Mgr Basic Fashion Dolls
Ken Smith	kens	Business Admin, Ad Planner, Merch Planner, Cat Mgr (all), Executive
Tony Jones	tonyi	Ad Planner, Merch Planner, Cat Mgr (all)

Promotion History

The sample dataset includes promotion history for the following time periods. These are the time periods that can be used when using the sample Promo Intelligence reports.

Transaction History: 12/29/2002 - 04/05/2003, 12/28/2003 - 4/10/2004,

and 1/2/2005 - 12/31/2005

Promo History: 1/31/2003 - 3/1/2003, 2/1/2004 - 3/6/2004 and 1/2/2005 - 3/13/2005

Focus Periods

By default, the sample dataset includes predicated baseline windows as follows. These are the only time periods that support forecasting and predictive analytics for this dataset.

Fiscal Week Number	Start Date
49	12/31/2006
50	1/7/2007
51	1/14/2007
52	1/21/2007
1	1/28/2007
2	2/4/2007
3	2/11/2007
4	2/18/2007
5	2/25/2007
6	3/4/2007
7	3/11/2007
8	3/18/2007
9	3/25/2007

Prediction Candidates

The following table describes which items in the dataset have been configured to support prediction. The item list is a function of which categories have been modeled, what sample transaction data exists, etc.

Table A–3 Items to Support Predictions

Item ID	Item Description	Class Name	Image
T0000048981	BB Beautiful Bride	Basic Barbie Dolls	Barbiebride.jpg
T0000048458	BB C.G. Horse Doll Ast.	Basic Barbie Dolls	barbiecghorsedoll.jpg
T0000048377	BB Cali Girl Barbie	Basic Barbie Dolls	barbiecaligirl.jpg
T0000049015	BB Happy Birthday	Basic Barbie Dolls	
T0000049031	BB Pet Doctor	Basic Barbie Dolls	barbiepetdoctor.jpg
T0000049082	BB W Kitchen	Basic Barbie Dolls	
T0000049039	BB W Nursery	Basic Barbie Dolls	
T0000100045	Pregnant Midge	Basic Barbie Dolls	barbiepregnantmidge.jpg
T0000071867	Ants In The Pants	Hasbro Preschool Games	antsinthepants.jpg
T0000051222	Boohbah Game	Hasbro Preschool Games	boohbah.jpg
T0000051769	Buckaroo	Hasbro Preschool Games	buckaroo.jp
T0000071859	Candyland	Hasbro Preschool Games	candyland.jpg
T0000855189	Don T Wake Daddy	Hasbro Preschool Games	dontwakedaddy.jpg
T0000246444	Hide & Seek	Hasbro Preschool Games	hideandseek.jpg

Table A–3 (Cont.) Items to Support Predictions

Item ID	Item Description	Class Name	Image
T0000961305	What A Mole	Hasbro Preschool Games	whacamole.jpg
T0000875341	Wp Pooh Candyland	Hasbro Preschool Games	poohcandyland.jpg
T0000361057	Pk Southern Islands Box Set	Pokemon Ccg	

Using the KSInc Dataset

This section describes using the KSInc dataset.

Installing the dataset

The dataset is installed using the a script located in the $PCE_HOME/sample directory. Running the deploy.sh script with no arguments will list the available options. Use the all option to install the entire dataset.

```
bash$ bash deploy.sh all
```

Creating Promotions

When creating promotions, note the existence of two sample templates. TheCircular - Std Weekly is an example of a template that uses a circular and 6 positioned pages. The Ad Hoc Promo template illustrates how to create an un-positioned page. When using Promotion Planning for the first time, use the Circular - Std Weekly template as it illustrates a more complete set of capabilities.

B

Managing Your Applications

This appendix provides reference material for systems administrators and others responsible for maintaining the Promotion Intelligence and Promotion Planning and Optimization production hosting environment. It contains the following sections:

Locating Host Specific Parameters

The following files contain host specific settings. Many of these default to "localhost" but may be different in a production environment.

```
$installdir/config/SIT/sit-config.properties
$installdir/config/promote/promote.properties
$installdir/modules/tools/bin/storesetupdater.sh
$installdir/modules/tools/conf/jndi.properties
$installdir/modules/tools/conf/promote-cmdline.properties
$installdir/modules/pce/sample/deploy.sh
$installdir/modules/pce/sample/images/install_is.sh
$installdir/config/SIT/server-list.xml
```

Monitoring the Promote Server

Besides the existing Oracle Application Server for monitoring facilities for monitoring applications, the following URL can be used to provide a pulse check of the web component to ensure that it is running properly:

```
http://${WL_HOST}:${WL_PORT}/promote/service.do?command=status
```

If everything is running properly, this command will return the string "OK"

Monitoring Policy Central Enterprise (PCE) Servers

To monitor the PCE servers, execute the following Linux command on each PCE host:

```
bash$ pceserver.sh -query status
Querying PCE server status....
PCE server is up running.
```

If the PCE server is not responding, use the following command to retrieve the server load.

```
bash$ pceserver.sh -query load
Querying PCE server load....
PCE server is running 0 applications.
```

For a listing of additional PCE server tools, run the pceserver.sh command with no options. A list of available commands will be provided.

Monitoring Batch Jobs

The following section provides an introduction to batch jobs.

Overview

Batch operations are composed of tasks, steps, and agents.

- Tasks – tasks are aligned with the major batch operations. These must be scheduled using a management tool and typically are enabled on one host.
- Steps – each task is composed of a number of steps, such as waiting for a file, unzipping it, and processing it.
- Agents – tasks and steps are managed as a process outside the application server. Agents run within each application server.

Tasks

The following are the tasks currently configured in the system:

- weekly.load – this task is responsible for loading and processing the appropriate data passed to the application each week.
- weekly.restate – this task also runs weekly, but it is meant specifically for certain data files separate from weekly.load.
- nightly.load – running nightly, this task loads the data that needs to be processed on a nightly basis.
- nightly.export – this task also runs nightly and is responsible for preparing data for export back to the client.

Steps

Each task defines a list of steps required to execute the task. This list of steps is task specific and can be found within $AUTOMATION_BASE/config/<task>/Proccess.steps.

The Job Log Table

Weekly and nightly feeds log their activities to the PR_MGMT_TASK table. The table below lists the columns in this table.

Table B–1 PR_MGMT_TASK Table Attributes

Column Name	Data Type	Nullable	Data Default	Column ID	Primary Key	Comments
RUN_ID	Number (32,0)	No			1	
TASK_NAME	Varchar2 (40 byte)	No			2	
TASK_ID	Number (32,0)	Yes			3	
STEP_NAME	Varchar2 (80 byte)	No			4	
STEP_START	Date	No			5	
STEP_END	Date	Yes			6	
STEP_ DURATION	Number (5.0)	Yes			7	
STEP_RESULT	Number (3,0)	No			8	
HOST	Varchar2 (200 byte)	No			9	
ARGUMENTS	Varchar2 (2048 byte)	Yes			10	
DESCRIPTION	Varchar2 (2048 byte)	Yes			11	

The application writes to this table with each step of its nightly or weekly batch cycle. The following is a brief example of what is written:

Table B–2 Sample Nightly Batch Data within the PR_MGMT_TASK Table

RUN_ID	TASK_ NAME	TASK_ID	STEP_NAME	STEP_RES	HOST	DESC
1	nightly.load		start	0	dev-app-101	Michaels Arts and Crafts: START in nightly.load (Run ID 20071018-1312) Automation log file is at ../mdc/operations /logs/automation /nightly.load. 200710181312.log All the logs are at ../mdc /operations/logs
2	nightly.load	1	generic/ wait.for.files.sh	0	dev-app-101	Running step 1/21 (generic/wait.for. files.sh) Step ended

Table B–2 (Cont.) Sample Nightly Batch Data within the PR_MGMT_TASK Table

RUN_ID	TASK_ NAME	TASK_ID	STEP_NAME	STEP_RES	HOST	DESC
3	nightly.load	1	update	0	dev-app-101	Beginning nightly processing...
...						
39	nighly.load	1	promote/ load/data.sh	0	dev-app-101	Running step 20/21 (promote/load.data.sh) Step ended
40	nightly.load	1	update	0	dev-app-101	Nightly load complete
41	nightly.load	1	end	0	dev-app-101	Michaels Arts and Crafts: SUCCESS in nightly.load (Run ID 200710181312)

As this example shows, each task is made up of many steps linked together using the TASK_ID column. Each step receives one row in the above table.

Understanding Job Failures

The critical column to watch is STEP_RESULT. If the step succeeds, there will be a zero in this column. A non-zero value represents a failure.

Operators should be alerted when a step fails. The following table illustrates an example failure.

Table B–3 Sample Task Failures within the PR_MGMT_TASK Table

RUN_ID	TASK_ NAME	TASK_ID	STEP_NAME	STEP_RES	HOST	DESC
1	weekly.load		start	0	dev-app-101	Michaels Arts and Crafts: START in nightly.load (Run ID 20071024-1517) Automation log file is at ../mdc/operations /logs/automation /weekly.load. 200710241517.log All the logs are at ../mdc /operations/logs

Table B–3 (Cont.) Sample Task Failures within the PR_MGMT_TASK Table

RUN_ID	TASK_ NAME	TASK_ID	STEP_NAME	STEP_RES	HOST	DESC
2	weekly.load	1	generic/ wait.for.files.sh	0	dev-app-101	Running step 1/21 (generic/wait.for. files.sh) Step ended
3	weekly.load	1	fail	2	dev-app-101	Michaels Arts and Crafts: FAILURE in weekly.load (RUN ID 200710241517)

The log file above demonstrates that the client failed to deliver the data files in the specified time window:

```
2007.10.24 15:17:17 Started processing weekly.load on 200710241517 with log at
../mdc/operations/logs/automation/weekly.load.200710241517.log.
2007.10.24 15:17:18 Running step generic/wait.for.files.sh (1/38):
2007.10.24 15:17:18 Step started at 2007.10.24 15:17:18.
2007.10.24 15:18:33 Have NOT received 'weekly.ash_lh_tbl.txt.gz'.
2007.10.24 15:18:33 Have NOT received 'weekly.ash_mh_tbl.txt.gz'.
2007.10.24 15:18:33 Have NOT received 'weekly.stage_mh_attrs_tbl.txt.gz'.
2007.10.24 15:18:33 Have NOT received 'weekly.bee_pr_like_merchandise.txt.gz'.
2007.10.24 15:18:33 Have NOT received 'weekly.bee_promo_offer_attr.txt.gz'.
2007.10.24 15:18:33 Have NOT received 'weekly.bee_promo_offer_merch.txt.gz'.
2007.10.24 15:18:33 Have NOT received 'weekly.bee_promo_offer.txt.gz'.
2007.10.24 15:18:33 Have NOT received 'weekly.bee_promo_store.txt.gz'.
2007.10.24 15:18:33 Have NOT received 'weekly.bee_promotions.txt.gz'.
2007.10.24 15:18:33 Have NOT received 'weekly.wk_hist_sales_inv.txt.gz'.
2007.10.24 15:18:33 Have NOT received 'weekly.bee_mb_detail.txt.gz'.
2007.10.24 15:18:33 Have NOT received 'weekly.data_ready.txt'.
2007.10.24 15:18:33 FATAL: Files are late and abort time has passed. Giving up.
2007.10.24 15:18:33 Step ended at 2007.10.24 15:18:33. Elapsed time is 75 seconds.
2007.10.24 15:18:34 Running procedure 'pr_mgmt.task_fail' with parameters '223, 2
, 'Demo Stores: FAILURE in weekly.load (Run ID 200710241517)''
2007.10.24 15:18:34 Ran procedure 'pr_mgmt.task_fail' successfully.
```

Locating Bad Data

A common failure is bad data received from the client. This may come across several ways. Suppose the client mistakenly puts a header row on the input feed. The following table is a sample task log illustrating the failed step.

Table B–4 Sample Bad Data Example

RUN_ID	TASK_ NAME	TASK_ID	STEP_NAME	STEP_RES	HOST	DESC
239	weekly.load	226	start	2	dev-app-101	Running step 13/38 (db/oracle/stage.file .sh Step ended

Using SQL Developer, the failed file can be located as follows:

The failed step was a result of a bad weekly.bee_promotions.txt file. Open the log file, and scan it for the failed task as illustrated in the example below:

```
2007.10.24 16:14:35 Running step db/oracle/stage.file.sh (13/38):
2007.10.24 16:14:35 Step started at 2007.10.24 16:14:35.
2007.10.24 16:14:35 Staging logs at
'../mdc/operations/logs/database/stage/weekly.bee_
promotions.txt.200710241614.log'.
2007.10.24 16:14:35 Bad records file at
'../mdc/operations/logs/database/stage/weekly.bee_
promotions.txt.200710241614.bad'.
2007.10.24 16:14:36 Staging file
'../mdc/operations/spool/automation/weekly.load/weekly.bee_promotions.txt' with
control file
'../install/modules/Database/BEECHSchema/install/oracle/BEECHSchema/controlfiles/b
ee_promotions.ctl'...

Load completed - logical record count 23.
2007.10.24 16:14:36 FATAL: sqlldr exited with an error.
2007.10.24 16:14:36 Step ended at 2007.10.24 16:14:36. Elapsed time is 1 seconds
```

The log files above demonstrate that the failure was a number format error and the actual offending record.

Restarting Jobs

Failed weekly batch jobs can be restarted using the $AUTOMATION_BASE/scripts/do.weely.load script.

Jobs can also be restarted at a certain step by providing the line number of the job to the task script. For example, running $AUTOMATION_BASE/scripts/do.weekly.load 7 would run weekly batch starting at step 7 assuming that the files have already been placed in their processing/spool directory.

Model Migration

To migrate a model that has been built from a development system to a production system do the following:

1. Capture the models to be migrated. Process them into intermediate files (.pmmlmv) using the script /install/modules/pce/bin/capture_model.sh.
2. Use a manual process to move the intermediate files from the source system to the target system.

3. Import the intermediate files in the target system and convert them into model files (.pmml). Deploy these migrated models using /install/modules/pce/bin/import_model.sh.

Parallel Processing

The two supported ways to run any process in parallel are:

- Manually run more than one process
- Run more than one process in operations

The Master Script ensures that the tasks identified as part of the configured SQL Templates are registered and executed only after the required pre-processing is complete. The Helper Process ensures that the master has registered the task. If not, it polls for the task in the cron process and the manual process exits without processing the task.

Manual Process

The following generic script acts as helper for the master:

process\scripts\common\process_exec_helper_manual_doit.sh.yapp.

This script is only responsible for executing the tasks in parallel. It checks the task table to identify if any process among the registered tasks is waiting for execution. If such a task is present, it executes the task or else terminates.

For example:

For lift models, the 1.build_lift_model_manual_helper.sh resides under \installdir\modules\pce\bin\build_lift_model. This script executes tasks in parallel with the Main Process. This ensures that only Master Scripts are responsible for registering the list of the tasks for that process and that helpers are only responsible for selecting the task each time.

Running More Than One Process in Operations

The following script should be used when a task are executed as a cron job. The cron job for the helper process should be scheduled to execute a few minutes before the master cron job. The following generic script is added that acts as helper for the master:

process \scripts\common\process_exec_helper_doit.sh.yapp. This script is only responsible for executing the task runs in parallel. It checks the task table to identify if any process is pending to execute the registered task for the specific process; if present it executes them. Helper process wakes every few minutes to determine if any registered process can be selected.

This script is only responsible for executing the task runs in parallel. It checks the task table to identify if any process among the registered tasks is waiting for execution. If such a task is present, it is executed. The helper process wakes every few minutes to determine if any registered process can be selected.

For example:

The Tae process has a master and helper script:

Master: \mdc\operations\automation\step.lib\promote\pce\tae.sh

Helper (internally calls process_exec_helper_doit.sh): \mdc\operations\automation\step.lib\promote\pce\tae_helper_run.sh

The operation configuration steps for the automation scripts mdc\operations\automation\config are:

- Master – weekly.load
- Helper – weekly.helper.load

The master scripts for the model build are:

- Main product – build_lift_model.sh \installdir\modules\pce\bin\
- MDC – doit.sh \mdc\scripts\6.build_lift_models\4.build_and_measure

The master scripts for tae are:

- Main product – tae_run.sh \installdir\modules\pce\bin
- MDC – doit.sh \mdc\scripts\4.analyze_history\2.tae

The updates to the SQL templates for customization are:

For the operations that Tae Runs uses:

tae_run_sql_template.txt residing under \mdc\operations\automation\step.lib\promote\pce\tae.

For the operations MDC Tae Runs uses:

tae_run_sql_template.txt residing under \mdc\scripts\4.analyze_history\2.tae.

Any updates to this template affect the tae runs in the main MDC scripts.

Other Maintenance Activities

The following are additional steps that can be taken to ensure proper maintenance of the production environment.

Starting and Stopping the Promote Server

Use the standard Oracle Application Server facilities to accomplish this.

Starting and Stopping the PCE

Use the following pceserver.sh commands to control and monitor the PCE server:

```
$bash pceserver.sh -start
$bass pceserver.sh -shutdown
```