Oracle® Retail Price Optimization

Operations Guide

Release 12.0

July 2006

ORACLE®

Price Optimization Operations Guide, Release 12.0

Contents

3 Standard Load

4 The Model Run

Preface

Price is an application that provides markdown recommendations and forecasts that allow customers to make informed markdown decisions. In this way, customers can maximize gross margins on seasonal merchandise while clearing inventory to specified levels by defined dates.

Audience

This document is intended for system administrators who configure and manage Price.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, see the following documents in the Oracle Retail Price Optimization Release 12.0 documentation set:

- *Oracle Retail Price Optimization Installation Guide*
- *Oracle Retail Price Optimization User Guide*
- *Oracle Retail Price Administration Guide*
- *Oracle Retail Price Optimization Configuration Guide*
- *Oracle Retail Price Optimization Operations Guide*
- *Oracle Retail Price Optimization Release Notes*

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 the Price Operations Guide

The Price Operations Guide provides details about the essential tasks involved in using Price: the staging and loading of data that is provided by the customer in specified formats and the weekly processes that produce markdown recommendations and forecasts.

What's In This Book

The Price Operations Guide addresses the following topics:

- Chapter 1 - Introduction - an orientation to the Price Operations Guide.
- Chapter 2 - Standard Interface - explains the data format specifications for the customer-supplied data feeds to Price.
- Chapter 3 - Standard Load - describes the load procedures for loading the data feeds into Price.
- Chapter 4 - Model Run - describes all the commands involved in the weekly batch process and the optimization run.
- Chapter 5 - RDM Load - describes the RDM refresh loads.
- Chapter 6 - Tools - describes the PriceAdmin tools.
- Chapter 7 - Troubleshooting - guidelines for fixing problems and a list of error messages.

2

Standard Interface

This chapter contains the following:

Introduction

An important part of getting Price up and running in a production environment is the gathering and loading of enterprise data. Price requires historical and weekly data to be loaded into the Price 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 Price.

Price Standard Interface Descriptions

This section details the data interface to the Price application. Price requires that customer data be provided in flat files containing pipe-delimited data organized so that the data can be loaded into Price database tables that follow the formats specified here.

For DB2, each data load file must end with a pipe delimiter. For Oracle, the terminal pipe delimiter is optional, but recommended.

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 Price 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 Price configuration. More information on these three interfaces is provided in Chapter 3, "Standard Load."

The standard interface includes the following:

Table 2–1 Interface Specifications

Interface Specification	Required/Optional
Merchandise Hierarchy	Required
Location Hierarchy	Required
Calendar	Required
Items	Required
Sales/Inventory/Orders	Required
Markdowns Taken	Required
Budget	Optional
Promotions	Optional
Distribution Center Inventory	Optional
Distribution Center Allocation	Optional
Merchandise Hierarchy Rename	Optional
Location Hierarchy Rename	Optional
Merchandise Hierarchy CDA	Optional
Location Hierarchy CDA	Optional
Items CDA	Optional
Business Rule Instances	Optional
Demand Parameters	Required
Price Ladders	Required
Seasonalities	Required
Cluster Definition	Optional
Cluster Mapping	Optional

Merchandise Hierarchy Standard Interface

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 2–2 on page 4, 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.

How Price Uses the Merchandise Hierarchy Data

Price uses the merchandise hierarchy data in a variety of ways, such as:

- To define worksheets, such as at the Department level
- To allow business rules to be assigned at higher levels than the item level

- To aggregate metrics in the Price UI and in Price reports
- To filter data in the Items Worksheet

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 Price end user. It may be displayed in the Price UI. 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. The key may not be displayed in the UI; however, 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.

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 2–2 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, Price can automatically detect and handle hierarchy moves without additional data. For more information on how Price manages merchandise

hierarchy changes, see "Merchandise Hierarchy Rename Standard Interface" on page 17.

- 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 2–2 on page 4, 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 lowest level does not have to be the level at which Price optimization (called the "item level") occurs; however, the lowest level typically is the optimization level.
- 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.

Location Hierarchy Standard Interface

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 on page 6, each record describes the region and company of a specific store.

How Price Uses the Location Hierarchy Data

Price uses the location hierarchy data in a variety of ways, such as

- To allow business rules to be assigned at higher levels than the item
- To aggregate sales data to the item level (for example, sales are at store level; items are at region level)
- To aggregate metrics in the Price UI and in Price reports
- To filter by location hierarchy in the Items Worksheet

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 Price end-user. It may be displayed in the UI. 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. The key may not be displayed in the UI; however, 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 2–3 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, Price can automatically detect and handle hierarchy moves without additional data. For more information on how Price manages location hierarchy changes, see "Location Hierarchy Rename Standard Interface" on page 17.
- 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 2–3 on page 6, 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 lowest level does not have to be the level at which Price optimization (called the "item level") occurs. The lowest level in the location hierarchy is typically the store level. The item level is often higher.

- 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.

Calendar Standard Interface

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

How Price Uses the Calendar Data

Price uses the calendar data in a variety of ways, such as:

- To construct the markdown calendar that defines the valid markdown effective dates.
- To determine in what month the markdown effective date falls. The markdown effective month affects metrics displayed in the Price UI and in Price reports such as "Markdown Budget."

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 2–4 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 Price. 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. The season could then be used in Price metrics such as "Season to Date Sales."

Items Standard Interface

The items interface describes valid combinations of merchandise and location that specify a Price item. All items in the Price system 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 3, "Standard Load.")

How Price Uses the Items Data

Price uses the items data in a variety of ways, such as:

- To define the total set of valid items for markdown optimization. (Some items are typically excluded each week, based on their eligibility.)
- To define key fields that affect the determination by Price of the item's seasonality and model start date. (The model start date defines the date when sales are included in the Price calculation of forecasts and markdowns.)

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 in Price for an item. Several Price metrics, such as Model Start Date and First Sale Date, are lower bounded by First Receipt Date.
- LAST_RECEIPT_DATE - the date of the most recent receipt of this item at the item's location. This date is only used for metrics in the Price UI.
- VENDOR - the supplier for the item.
- VENDOR_DESC - a description of the supplier.
- UNIT_COST - the average unit cost of the item. This data is used by Price for margin calculations (metrics) only. These calculations do not affect the forecast and markdown recommendations made by Price.
- 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 2–5 Items Sample Data

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

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.

Technical Notes

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

- All items must be defined at the same level of the merchandise hierarchy and location hierarchy. Because of this, it may not be possible, for example, to define some items at the Chain level of the location hierarchy and to define other items at the Region level.
- Price loads, aggregates (if necessary), and persists sales and distribution center inventory data only at the item level.
- Retailers may start sending weekly data to Price for items that are not initially available for markdown optimization (if, for example, a retailer only turns on optimization for a subset of the business). Price accumulates the sales history for items that may then be made available for markdown optimization in the future.
- The lowest level for an item in the Price merchandise hierarchy may not correspond to the lowest level in a retailer's merchandise hierarchy. A retailer may need to aggregate some of the data provided in the items interface. For example, if the lowest level in the Price merchandise hierarchy is color, but the retailer has size beneath color, then the retailer will need to aggregate the items data fields as follows:
  - FIRST_RECEIPT_DATE - minimum aggregation
  - LAST_RECEIPT_DATE - maximum aggregation
  - UNIT_COST - average or weighted average, based on the total inventory aggregation

Sales/Inventory/Orders Standard Interface

The sales interface describes weekly sales, inventory, and order data at the lowest level of the merchandise and location hierarchy. If items are defined at higher levels in either hierarchy, then Price aggregates the data in the sales file to the level required by the items.

For the weekly data, the sales file should always contain the sales records for the most recent week of sales that have not yet been sent to Price. The same weekly sales file can also contain the complete sales records for one or more previous weeks of sales (for example, in order to correct previous weekly sales data because of subsequent adjustments). When the sales file contains data for previous weeks, it must contain the complete sales data for those weeks, not just the changes. The load procedure replaces the data already loaded with the new data.

How Price Uses the Sales Data

Price uses the sales data in a variety of ways, such as:

- To define the current selling price for an item in the absence of any promotions.
- To determine, in combination with analytical parameters, the base demand for an item.
- To calculate a number of historical sales and inventory-related metrics.
- To help define the total inventory to clear through markdown optimization (inventory on-hand is always part of the total inventory to clear and units on-order are typically part of the total inventory to clear provided by the model).

Data Fields

Sixteen fields describe each sales record:

- MERCHANDISE_KEY - the key for the lowest level in the merchandise hierarchy associated with this sale.
- LOCATION_KEY - the key for the lowest level in the location hierarchy associated with this sale.
- FISCAL_YEAR - the fiscal year of the sales record.
- FISCAL_WEEK - the fiscal week of the sales record.
- NET_SALES_UNITS - the number of units sold minus the number of units returned for the merchandise at the location. Price uses the net sales to calculate demand and to calculate sales-related metrics shown in the Price application.
- NET_SALES_DOLLARS - the dollars received at the register for all new sales minus the return dollars. Net sales dollars are used to calculate sales-related metrics shown in the Price application.
- GROSS_SALES_UNITS - the number of units sold, excluding returns, for the merchandise at the location. Gross sales units are used, in conjunction with GROSS_SALES_DOLLARS, to calculate the sales price that is passed to the Price model and used to calculate demand. Gross sales are not used for sales-related metrics shown in the Price application.
- GROSS_SALES_DOLLARS - the dollars received at the register for all new sales, excluding returns. Gross sales dollars are used, in conjunction with GROSS_SALES_UNITS, to calculate the sales price that is passed to the Price model and used to calculate demand. Gross sales are not used for sales-related metrics shown in the Price application.

- POS_SALES_UNITS - the number of units sold at a promotional price (temporary markdown taken at the register) for the merchandise at the location.
- POS_SALES_DOLLARS - the dollars received at the register for the POS_SALES_UNITS.
- EOP_INVENTORY_UNITS - the number of units of the merchandise on hand at the location at the end of the fiscal week.
- EOP_ON_ORDER_UNITS - the number of units of the merchandise on order for or in transit to the location at the end of the fiscal week.
- CURRENT_RETAIL - the retail sales price of the merchandise at the location at the end of the week in the absence of any promotional discounts. Current Retail is used as the starting price for Price forecasts and optimizations. CURRENT_RETAIL should not reflect planned changes to the retail sales price in the coming week or weeks. (Such pending changes are specified in the Markdowns Taken interface, which is described in "Markdowns Taken Standard Interface" on page 12.)
- CURRENT_INV_PRICE - the inventory price of the merchandise at the location at the end of the fiscal week. This price is used as the basis for calculating markdown dollars, using the retail accounting method. Some retailers may use a retail accounting method in which the inventory price of an item is not always the same as the retail sales price.
- STORE_NUM_WITH_INV - the number of locations that have positive inventory (EOP_INVENTORY_UNITS) at the end of the fiscal week. Assuming the sales records are at the store level in the location hierarchy, this value will be equal to one when EOP_INVENTORY_UNITS is greater than zero, and it will be equal to zero when EOP_INVENTORY_UNITS is less than or equal to zero.
- STORE_NUM_WITH_OO - the number of locations that have positive units on order, but zero units on hand, at the end of the fiscal week. Assuming the sales records are at the store level in the location hierarchy, this value will be equal to one when EOP_ON_ORDER_UNITS is greater than zero and EOP_INVENTORY_UNITS is less than or equal to zero, and it will be equal to zero in all other cases.

An Example

The following table shows sample data, using previously defined items.

Table 2–6 Sample Sales Data

Merch. Key/Loc. Key	Fisc. Yr.	Fisc. Wk.	Net Sales Units	Net Sales $	Gross Sales Units	Gross Sales $	POS Sales Units	POS Sales $	EOP Inv. Units	EOP OO Units	Store # Inv.	Store # OO	Curr. Rtl. Price	Curr. Inv. Price
101234509 /1000	2004	52	3	69.97	3	69.97	0	0.00	41	0	1	0	24.99	24.99
101234509 /1001	2004	52	4	74.96	5	99.95	0	0.00	60	0	1	0	19.99	19.99
101234509 /1010	2004	52	1	19.99	1	19.99	0	0.00	0	20	0	1	19.99	19.99
101234512 /2000	2004	52	2	84.98	2	84.98	1	39.99	28	0	1	0	49.99	49.99

Notes on Table:

- Row 1: The Net Sales Dollars are the actual sales dollars captured at the register and can thus be less than the Net Sales Units multiplied by the Current Retail Price (even if there are no POS Sales Units).
- Row 2: The Gross Sales Dollars are not the same as the Gross Sales Units, because of one unit being returned.
- Row 3: This location has zero EOP Inventory Units and non-zero EOP On Order Units. Because of this, the value in the Store # OO is 1.
- Row 4: Two units were sold this week, one during a promotion and one outside a promotion.

Technical Notes

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

- Price only persists the sales data at the item level in the Price database. If a retailer requires a copy of the weekly sales data, she or he should implement a process to archive the weekly data.
- The gross and net sales fields are required by the interface. The POS sales fields can be NULL if not available. If a retailer cannot provide the gross sales fields, then those fields must be populated with the corresponding value for net sales.
- The CURRENT_INV_PRICE field is required by the interface. Retailers who do not have an inventory price that is distinct from the retail price should populate this field with the value in the CURRENT_RETAIL field.
- Price expects a single record in the sales feed to summarize the sales for a given merchandise/location/week. For example, if a retailer is sending sales to Price at the color-store level, they should send a single record for each color-store-week. The retailer is responsible for aggregating the transactions for the color-store-week to create this record.
- A retailer may have even lower levels in her or his merchandise hierarchy that are not known by Price; therefore, the retailer may need to aggregate some of the data provided in the sales interface. For example, if the lowest level in the Price merchandise hierarchy is color, but the retailer's hierarchy has size beneath color and the retailer's data source for sales is at the size level, then the retailer will need to aggregate the sales data fields as follows:
  - All UNITS and DOLLARS fields should be aggregated by summing the lower level records.
  - CURRENT_RETAIL and CURRENT_INV_PRICE should be aggregated by choosing the most frequently occurring value for the field. For example, if there are six sizes for an item (color), and five of the sizes have a current retail price of $19.99, and one of the sizes has a current retail price of $21.99, then the current retail sent to Price should be $19.99.
  - If the records in the sales file are at a lower level than the item level in Price, then Price will aggregate the sales data following the same aggregation rules just described.

Markdowns Taken Standard Interface

The markdowns taken interface describes permanent markdowns, past, present, or future, that have been entered into a retailer's price change execution system. Price supports one markdown for a given fiscal week.

The markdowns-taken records must be at the item level (the level used for optimization). For example, if Price markdown recommendations are at the color-region level, then the markdowns-taken records must be at the color-region level.

How Price Uses the Markdowns Taken Data

Price uses the markdowns taken data in a variety of ways, such as:

- To determine the markdowns that have already been executed in the store and the markdowns that are pending execution in the future. The CURRENT_RETAIL field in the Sales data feed does not reflect pending markdowns; however, Price builds the information about pending markdowns into its forecast. Price will not recommend new markdowns prior to any pending markdowns, but will recommend additional markdowns after pending markdowns if additional markdowns are needed and do not violate business rules.
- To determine the validity of future markdowns based on the number of previous markdowns and the date of the most recent markdown.

Data Fields

Eight fields describe each entry in the markdowns taken data:

- MERCHANDISE_KEY - in combination with the location key, identifies the item being marked down
- LOCATION_KEY - in combination with the merchandise key, identifies the item being marked down
- EFFECTIVE_DATE - the expected store execution date of the markdown
- PRICE_VALUE_TYPE - Prices are expressed as either Percentage Off Original Retail Price (PO), Percentage Off Ticketed Price (PT), or Price Point (PP)
- ACCOUNTING_TYPE - The acounting type for the markdown can be either Permanent (PERM) or Temporary (TEMP).
- PRICE_POINT - If PRICE_VALUE_TYPE is PP, then this contains the price point values. Either PRICE_POINT of PRICE_PCT_OFF must be not null, depending on the value in PRICE_VALUE_TYPE.
- PRICE_PCT_OFF - If PRICE_VALUE_TYPE is PO, then this contains the percentage off (a value between 0 and 1). Either PRICE_POINT of PRICE_PCT_OFF must be not null, depending on the value in PRICE_VALUE_TYPE.
- CLIENT_LADDER_ID - Unique identifer for the price ladder (i.e., unique per price ladder string)

An Example

The following table shows two sample markdowns taken records for the items from the previous examples.

Table 2–7 Sample Markdowns Taken Data

Merchandise Key	Location Key	Effective Date	Old Ticket Price	New Ticket Price
101234509	FL1	2005-02-07	24.99	19.99
101234509	FL2	2005-02-07	19.99	16.99

The first row in the table describes an item that is marked down from $24.99 to $19.99 in Price on Wednesday, February 2, 2005. The markdown will take effect in the stores on Monday, February 7, 2005. When the sales data feed is transmitted to ProfitLogic on Sunday, February 6, 2005, the item's CURRENT_RETAIL price in the sales file should still be $24.99, because that is the retail price as of the end of the week for the sales data feed. However, the item also has a record in the markdowns taken data feed that indicates that the item will be permanently marked down to $19.99 on Monday.

Technical Notes

The following list provides details to consider regarding the markdowns taken data.

- The markdowns taken data must be at the item level (the level of optimization). Retailers typically must aggregate markdowns taken data from lower levels. The markdowns taken aggregation should be consistent with the CURRENT_RETAIL aggregation described for the sales interface. Because of this, the retailer should generate a markdown taken record whenever there is a change in the most frequently occurring value of CURRENT_RETAIL.
- The new ticket price must be less than the old ticket price (the interface does not support markups). The effective date must be a day. No two records are allowed to have the same merchandise, location, and effective date values.
- Price is not the system of record for price changes. It is therefore strongly recommended that all retailers provide this data, even if the markdowns taken in Price are being automatically transmitted to the retailer's price execution system. In this way, Price is aware of any markdowns taken, either through Price itself or by some other means.

Budget Standard Interface

The budget interface describes markdown budget and other financial planning data. Each row in the data file contains the financial metrics for the level of the merchandise hierarchy that corresponds to the Price worksheet for a given month. The budget information is provided to help retailers understand the consequences of any markdown decisions.

How Price Uses the Budget Data

Price uses the budget data in a variety of ways, such as:

- to provide users with the budgetary context for their markdown decisions
- to derive an internal value for the Optimize to Budget screens

Data Fields

Here is some information about the key fields that specify the budget data.

- MARKDOWN_BUDGET
- PLANNED_GM_DOLLARS
- PLANNED_GM_PERC

Technical Notes

This file is optional, so retailers who do not want to see budget information in the Price user interface do not have to provide this data.

Promotions Standard Interface

The promotions interface describes planned promotions or temporary markdowns. Price uses this information to adjust forecasting and markdown recommendations.

How Price Uses the Promotions Data

Price uses the promotions data in a variety of ways, such as:

- To determine the item level forecast, which can affect markdown recommendations. For example, a permanent markdown may be delayed or may not be necessary because of a planned promotion.
- To restrict markdown recommendations. For example, a retailer can specify that a markdown should be taken only if it is deeper or more shallow than a certain type of planned promotion.

Data Fields

Here is some information about the key fields that specify the promotions data.

- PROMO_PRICE - Price during the promotion (a value must be provided for either PROMO_PRICE or PROMO_PERC_OFF)
- PROMO_PERC_OFF - the price is calculated as a percentage of the forecasted retail price at the time of the promotion. This must be a value between 0 and 1.
- PROMO_TYPE - The promotion type, or interpretation, is either ceiling (candidate prices can be no higher than the promotion price), floor (candidate prices can be no less than the promotion), or unrestricted.
- PROMO_EXCL_FG - Included (1) or excluded (-1) from a promotional event.

Technical Notes

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

- The promotion is specified as either a price (PROMO_PRICE) or as a percentage off (PROMO_PERC_OFF) the current retail price, so one of the two fields is always blank.
- If a promotion is specified as a percentage off, it must be expressed as a value between 0 and 1. (For example, 30% off is expressed as 0.3.).
- The promotions records must be at or above the item level (the level of optimization). For example, if Price markdown recommendations are at the region level, then the promotions records must be at the region level. If a promotion is above the item level, then the load will automatically explode the promotion down to the item level.
- The current week's promotion data should include all promotions that have an end date that occurs after the last date in the sales history, regardless of when the promotion started.
- The promotion type column (PROMO_TYPE) is mandatory.
- A point-of-sale (POS) promotion is always unrestricted.
- Price does not combine the price effects of simultaneous promotions on the same item. For example, if three promotions are in effect for an item on a given day, Price looks, in order, at the promotion's interpretation, priority, and price and determines the price. (The interpretation, or promotion type, and the priority, an

assigned value, are both defined in IR_PLANNED_PROMOS, which is described in the Price Configuration Guide.)

- If the price has already been marked down below the level of the promotion, then the current price does not change.
- When a promotion is defined and loaded, the promotion is applied (exploded) from the level at which it is defined down to the lower levels. The exclusion flag identifies merchandise within the defined promotion that is not to be included in the promotion.

The following table shows the use of the exclusion flag.

Table 2–8 Sample Promotion Data

Merch. Key	Merch. Level	Loc. Key	Loc. Level	Promo. Key	Start Date	End Date	Promo. Price	Prono. % Off	Promo. Desc.	Ex. Flag
10	DEPT	North	REGION	P1	01/01/04	02/01/04	null	20	Apromo	1
0	CHAIN	NE	DISTRICT	P1	01/01/04	02/01/04	null	20	Apromo	-1
1001	STYLE	North	REGION	P1	01/01/04	01/01/04	null	20	Apromo	-1
0	CHAIN	0	CHAIN	P2	02/14/04	02/15/04	null	null	PresDay	1
1002	STYLE	North	REGION	P3	01/01/04	02/01/04	null	20		1
1003	STYLE		CHAIN	*			null			-1

Notes on the table entries:

- Row 1 shows a month-long 20 percent off P1 promotion.
- Row 2 excludes all merchandise in the NE from the P1 promotion.
- Row 3 excludes the 1001 Style from the P1 promotion from the entire North Region. (Row 2 simply excluded all merchandise from the NE.)
- Row 4 shows a non-price promotion over President's Day for the entire company.
- Row 5 shows a month-long 20 percent off P3 promotion.
- Row 6 excludes Style 1003 from every promotion.

Distribution Center Inventory Standard Interface

The distribution center inventory interface describes inventory and merchandise on order for a given distribution center.

Distribution Center Allocation Standard Interface

The distribution center allocation interface describes how merchandise is to be allocated to the locations supplied by a warehouse. DC inventory allocation information can be supplied at any level of the merchandise hierarchy.

Technical Notes

The following list provides details to consider regarding the dc allocation data.

- The data includes the proportion (fraction) of merchandise that should be allocated to each location.
- Fractional units are truncated.
- Fractions can be specified at any level of the merchandise hierarchy.

- Fractions can only be specified at the location hierarchy optimization level.
- A fraction at a lower level always takes precedence over a fraction at a higher level.

Merchandise Hierarchy Rename Standard Interface

The merchandise hierarchy rename interface facilitates reclassifying and moving merchandise within the merchandise hierarchy. 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.

Location Hierarchy Rename Standard Interface

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 CDA Standard Interface

The merchandise hierarchy cda interface provides 24 additional optional attributes. For more information, see the Configurable Data Attributes chapter.

Location Hierarchy CDA Standard Interface

The location hierarchy cda interface provides 24 additional optional attributes. For more information, see the Configurable Data Attributes chapter.

Items CDA Standard Interface Specification

The items cda interface specification provides the following 24 additional optional attributes. For more information, see the Configurable Data Attributes chapter.

Business Rule Instances Standard Interface

The data to be loaded by the Business Rule Manager bulk loader utility must conform to the following standard interface specification. For more information on the Business Rule Manager, see the Price Configuration Guide.

Technical Notes

The following list provides details to consider regarding the business rule instance data.

- The merchandise and location keys map to the CLIENT_LOAD_ID.
- The merchandise and location levels map to LEVEL_DESC.
- The rule name is the name of the business rule as specified in the business rule definition.
- The rule value is the value assigned to the business rule instance.
- The attribute values are the specific values for the custom variables, which have been derived from columns in the permitted source tables.
- The delete flag defines whether the instance is to be deleted (a value of 1) or added/updated (a value of 0 - the default).

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.

How Price Uses the Parameters Data

Price uses the parameters data in a variety of ways, including:

- To provide a centralized list for the parameters and their values

  AS_PARAMETER_ID and AS_VERSION_NUMER are used only by Analytical Services; they are not used by Price.

Data Fields

Nine fields describe each 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.

Price Ladders Standard Interface

The price ladders standard interface describes the price ladders displayed in the Price UI.

Note: The Price ladders data feed will not load if the data contains a colon or a semi-colon.

How Price Uses the Price Ladder Data

Price ladders define a client-specific set of markdown prices that can be selected in the Price application. Prices in the price ladder are expressed either as a price point (PP), as a percentage off the original retail price (PO), or as a percentage off the ticket price (PT). Each of these three types of price ladder can be permanent or temporary. Temporary price ladders are denoted by a t- prefix in the UI.

Data Fields

Twelve fields describe each price ladder:

- CLIENT_LADDER_ID - An externally generated sequential number that identifies the price ladder.
- MERCHANDISE_KEY - the key for this level of the merchandise hierarchy.

- MERCHANDISE_LEVEL - the level of the merchandise hierarchy.
- LOCATION_KEY - the key for this level of the location hierarchy.
- LOCATION_LEVEL - the level of the location hierarchy.
- PRICE_VALUE_TYPE - price ladders are expressed as either percentage off original retail price (PO), percentage off ticketed price (PT), or price point (PP).
- PRICE_LADDER_DESC - The price ladder name that is displayed in the UI.
- MODEL_FLAG - A model run indicator. “R” indicates that the price ladder is used for the optimization.
- ITEM_PRG_FLAG - Indicates if the price ladder is used for the optimization of individual items only (ITEM), pricing groups only (PRG), or both (BOTH).
- ACCOUNTING_TYPE - The accounting type can be either permanent (PERM) or temporary (TEMP).
- PRICE_POINT - If the price ladder type is PP, then this contains the price point values. Either PRICE_POINT or PRICE_PCT_OFF must be Not Null, depending on the value of the PRICE_VALUE_TYPE column.
- PRICE_PCT_OFF - If the price ladder type is PO, then this contains the percentage off (a value between 0 and 1). Either PRICE_POINT or PRICE_PCT_OFF must be Not Null, depending on the value of the PRICE_VALUE_TYPE column.

Technical Notes

The following list provides details to consider regarding price ladder data:

- The price ladder data is generally loaded after the merchandise/location hierarchy information, since price ladders are tied to levels. And, at least one price ladder must be loaded before the first model run.
- The inference rules, at a minimum, assign a default price ladder to each item or collection. In Price, ITEMS maps each ITEM_ID to a price ladder via the function getPriceLadderID. Each item can have only one price ladder for processing by the model.
- Percent off (PO) price ladders are discounts that are applied to the original price.
- Price point (PP) price ladders are actual prices that must be expressed as $x.99.

Price ladders define a client-specific set of markdown prices that can be selected in the Price application. Prices in the price ladder are expressed either as a price point (PP), as a percentage off the original retail price (PO), or as a percentage off the ticket price (PT). Each of these three types of price ladder can be permanent or temporary. Temporary price ladders are denoted by a t- prefix in the UI.

The price ladder information collected from a retailer should include the name of the price ladder, the type of ladder, a list of price points or percentages (the rungs), and the hierarchy/location level.

The price ladder data is generally loaded after the merchandise/location hierarchy information, since price ladders are tied to levels. And, at least one price ladder must be loaded before the first model run.

The inference rules, at a minimum, assign a default price ladder to each item or collection. In Price, ITEMS maps each ITEM_ID to a price ladder via the function getPriceLadderID. Each item can have only one price ladder for processing by the model.

Price Ladder Views

Price ladder information is displayed in the UI. Users select from the drop-down list of price ladder names and then select a price point or percentage from the drop-down list of price ladder values. For more information, see the Price User Guide.

Views in the UI are defined via the following:

P4P_LADDER_ROLES defines which price ladders are available for a specific worksheet.

Table 2–9 P4P_LADDER_ROLES

Column Name	Data Type	Nullable
HIERARCHY1	VARCHAR2 (50)	Y
HIERARCHY2	VARCHAR2 (50)	Y
HIERARCHY3	VARCHAR2 (50)	Y
HIERARCHY4	VARCHAR2 (50)	Y
LADDER_ID	Number	Y

P4P_LADDER_INFO identifies the name and type of each price ladder. It is recommended that price ladder names contain meaningful information, including price ladder type, to help users in making selections.

Table 2–10 P4P_LADDER_INFO

Column Name	Data Type	Nullable
LADDER_ID	Number (32,0)	N
LADDER_TYPE	Number	Y
LADDER_NAME	VARCHAR2 (50)	Y

P4P_LADDER_VALUES defines the rungs (fixed prices or percentages off) of the price ladder.

Table 2–11 P4P_LADDER_VALUES

Column Name	Data Type	Nullable
LADDER_ID	Number (32,0)	N
LADDER_VALUE	Number	Y
LADDER_TYPE	Number	Y

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 Price to calculate markdowns and forecasts.

How Price Uses the Seasonalities Data

Price uses the seasonalities data in a variety of ways, including:

- To support seasonality searches across the merchandise and location hierarchies.
- The following inference rules are involved in seasonalities:

- IR_SEASONALITIES - provides the seasonality values to the model from start date to out date.
- IR_SEASONALITY_ATTRIBUTE - defines the attributes value(s) used for seasonality matching.
- IR_ITEM_IDS - maps item IDs to seasonality IDs

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.

Cluster Mapping Standard Interface

The cluster mapping standard interface, part of Flexible Store Clustering, associates a cluster set with a pre-defined level in the merchandise hierarchy. Cluster sets are assigned to the merchandise hierarchy at the level specified by a system-wide configuration flag during the optimization process.

How Price Uses the Cluster Mapping Data

The cluster sets are assigned to the merchandise hierarchy. Individual clusters or cluster sets are used in the creation of worksheets and items. The clusters and cluster sets are stored in the location hierarchy. If a store cluster set is already assigned to a specified entry in the merchandise hierarchy, it will be replaced by the entry specified in the assignment element. Store clusters should not be linked to the merchandise hierarchy, but can be used in the creation of worksheets or items.

The merchandise key in the cluster mapping table must be at the merchandise level that is specified by the entry in ASH_CP_TBL for CLUSTER (INTERSECT_NAME).

Data Fields

Two fields describe cluster mapping:

- CLUSTER_SET_KEY - unique identifier of the cluster set, which is provided by the client
- MERCHANDISE_KEY - unique identifier from the Merchandise Hierarchy table.

An Example

The following table shows the assignment of cluster sets to the merchandise hierarchy:

Table 2–12 Cluster Mapping Sample Data

Cluster Set Key	Merchandise Key
CL1	1
CL1	2
CL2	3
CL2	99
MSA1	4
MSA1	6

Cluster Definition Standard Interface

The cluster definition standard interface, part of Flexible Store Clustering, defines cluster sets, clusters, and store to cluster mappings. The cluster sets and clusters are identified by their name or key. These names must be unique within the implementation in order to ensure that unique descriptions are displayed in the Price UI. The store location to cluster mapping is specified by the unique key, which is specified in the Location Hierarchy standard interface.

Data Descriptions

Five fields describe the cluster definition:

- CLUSTER_SET_KEY - unique identifier of the cluster set, which is provided by the client
- CLUSTER_SET_DESC - unique description of the cluster set
- CLUSTER_KEY - unique identifier of the cluster
- CLUSTER_DESC - unique description of the cluster
- LOCATION_KEY - unique indentifier from the Location Hierarchy table

An Example

The following table shows a cluster definition:

Table 2–13 Cluster Definition Sample Data

Cluster Set Key	Cluster Set Desc	Cluster Key	Cluster Desc	Store
CL1	CLIMATE1	CL1:COLD	COLD	288
CL1	CLIMATE1	CL1:WARM	WARM	280

Table 2–13 (Cont.) Cluster Definition Sample Data

Cluster Set Key	Cluster Set Desc	Cluster Key	Cluster Desc	Store
CL1	CLIMATE1	CL1:WARM	WARM	282
CL1	CLIMATE1	CL1:HOT	HOT	274
CL2	CLIMATE2	CL2:COLD	COLD	288
CL2	CLIMATE2	CL2WARM	WARM	280
CL2	CLIMATE2	CL2:HOT	HOT	282
CL2	CLIMATE2	CL2:HOT	HOT	274
MSA1	MSA1	MSA1:ATL	ATLANTA	280
MSA1	MSA1	MSA1:NYNY	NEWYORK	288
MSA1	MSA1	MSA1:NYNY	NEWYORK	282
MSA1	MSA1	MSA1:NYNY	NEWYORK	274

Cluster Levels Standard Interface (ASH_CSHL_TBL)

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

Technical Notes

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

- The x 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 cluster levels information is generally loaded only once.

Price Interface Specifications

The following tables provide ordered lists of the contents of each of the Price interface specifications.

Merchandise Hierarchy Specification (ASH_MH_TBL)

Table 2–14 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

Table 2–14 (Cont.) Merchandise Hierarchy Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
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
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

Table 2–14 (Cont.) Merchandise Hierarchy Standard Interface Specification

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
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

Location Hierarchy Specification (ASH_LH_TBL)

Table 2–15 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 2–15 (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

Calendar Specification (ASH_CAL_TBL)

Table 2–16 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

Items Specification (ASH_ITEMS_TBL)

Table 2–17 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
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.

Sales Specification (ASH_SALES_TBL)

Table 2–18 Sales/Inventory/Orders Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_KEY	Key for the lowest level in the merchandise hierarchy.	String	25	N
LOCATION_KEY	Key for the lowest level in the location hierarchy.	String	25	N
FISCAL_YEAR	Number of the fiscal year.	Integer	4	N
FISCAL_WEEK	Number of the fiscal week.	Integer	2	N
NET_SALES_UNITS	Describes the net number of units sold of the Merchandise at the Location.	Integer	22	N
NET_SALES_DOLLARS	Describes the net dollar amount of sales for the Merchandise/ Location during the fiscal week.	Decimal	22,3	N
GROSS_SALES_UNITS	Describes the gross number of new units sold of the Merchandise at the Location.	Integer	22	N
GROSS_SALES_DOLLARS	Describes the gross dollar amount of new sales for the Merchandise/ Location during the fiscal week.	Decimal	22,3	N
POS_SALES_UNITS	Describes the number of units of the Merchandise at the Location sold at a temporary markdown taken at the register.	Integer	22	Y
POS_SALES_DOLLARS	Describes the dollar amount of sales with a temporary markdown taken at the register for the Merchandise/ Location during the fiscal week.	Decimal	22,3	Y
EOP_INVENTORY_UNITS	Describes the number of units on hand inventory at the end of the fiscal week.	Integer	22	N
EOP_ON_ORDER_UNITS	Describes the number of units on order at the end of the period (in transit to the store)	Integer	22	N

Table 2–18 (Cont.) Sales/Inventory/Orders Standard Interface Specification[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
STORE_NUM_WITH_INV	Describes the number of locations that have inventory at the end of the fiscal week.	Integer	22	N
STORE_NUM_WITH_OO	Describes the number of locations that have units on order (but not on hand) at the end of the fiscal week.	Integer	22	N
CURRENT_RETAIL	Describes the merchandise's retail price.	Decimal	22,2	N
CURRENT_INV_PRICE	Describes the merchandise's inventory price.	Decimal	22,2	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.

Markdowns Taken Specification (ASH_MDTAKEN_TBL)

Table 2–19 Markdowns Taken Standard Interface Specification[1]

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_KEY	In combination with the location key, identifies the item being marked down	String	25	N
LOCATION_KEY	In combination with the merchandise key, identifies the item being marked down	String	25	N
EFFECTIVE_DATE	Effective date of the retail price change	Date in format YYYY-MM-DD	10	N
PRICE_VALUE_TYPE	Percentage Off Original Retail Price (PO), Percentage Off Ticketed Price (PT), or Price Point (PP)	String	2	N
ACCOUNTING_TYPE	The accounting type for the markdown can be either Permanent (PERM) or Temporary (TEMP).	String	4	N
PRICE_POINT	If PRICE_VALUE _TYPE is PP, then this contains the price point values. A value must be provided for either PRICE_POINT or PRICE_PCT_OFF.	Decimal	7,2	Y
PRICE_PCT_OFF	If PRICE_VALUE _TYPE is PO, then this contains the percentage off (a value between 0 and 1). A value must be provided for either PRICE_POINT or PRICE_PCT_OFF	Decimal	3,2	Y
CLIENT_LADDER_ID	Unique identifer for the price ladder (i.e., unique per price ladder string)	Integer		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.

Budget Specification (ASH_BUDGET_TBL)

Table 2–20 Budget Standard Interface Specification[1]

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_ KEY	Unique key to the merchandise hierarchy level of the worksheet	String	25	N
LOCATION_ KEY	Unique key to the location hierarchy level of the worksheet	String	25	N
FISCAL_YEAR	Fiscal Year ID	Integer	4	N
FISCAL_MONTH	Fiscal Month ID	Integer	2	N
MARKDOWN_BUDGET	Markdown budget	Decimal	22,3	Y
PLANNED_GM_DOLLARS	Planned GM Dollars	Decimal	22,3	Y
PLANNED_GM_PERC	Planned GM Percent	Decimal	22,3	Y
ATTRIBUTE1		Decimal	22,3	Y
ATTRIBUTE2		Decimal	22,3	Y
ATTRIBUTE3		Decimal	22,3	Y
ATTRIBUTE4		Decimal	22,3	Y
ATTRIBUTE5		Decimal	22,3	Y
ATTRIBUTE6		Decimal	22,3	Y
ATTRIBUTE7		Decimal	22,3	Y
ATTRIBUTE8		Decimal	22,3	Y
ATTRIBUTE9		Decimal	22,3	Y
ATTRIBUTE10		Decimal	22,3	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.

Promotions Specification (ASH_PROMO_TBL)

Table 2–21 Promotions Standard Interface Specification[1]

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_KEY	Key for this level in the merchandise hierarchy	String	25	N
MERCHANDISE_LEVEL	The level in the merchandise hierarchy under promotion	String	50	N
LOCATION_KEY	Key for this level in the location hierarchy	String	25	N
LOCATION_LEVEL	The level in the location hierarchy under promotion	String	50	N
PROMOTION_KEY	Some name that identifies the promotion under consideration	String	50	Y
PROMO_START_DATE	Start date of the promotion	Date in format YYYY-MM-DD	10	N
PROMO_END_DATE	End date of the promotion	Date in format YYYY-MM-DD	10	N
PROMO_PRICE	Price during the promotion (a value must be provided for either PROMO_PRICE or PROMO_PERC_ OFF)	Decimal	22,2	Y
PROMO_PERC_OFF	Percent off current retail price (a value must be provided for either PROMO_PRICE or PROMO_PERC_ OFF). Expressed as a value between 0 and 1.	Decimal	22,3	Y
PROMO_DESC	Description of the promotion	String	100	Y
PROMO_TYPE	Type of the promotion (floor, ceiling, or unrestricted). This column is mandatory.	String	50	N

Table 2–21 (Cont.) Promotions Standard Interface Specification[1]

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
PROMO_EXCL_FG	Included (1) or excluded (-1) from a promotional event.	Integer	2	Y
PROMO_NUMBER	Number of the promotion	Integer	4	Y
ATTRIBUTE1		String	50	Y
ATTRIBUTE2		String	50	Y
ATTRIBUTE3		String	50	Y
ATTRIBUTE4		String	50	Y
ATTRIBUTE5		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.

DC Inventory Specification (ASH_DCI_TBL)

Table 2–22 Distribution Center Inventory Standard Interface Specification

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_KEY	Key for the lowest level in the merchandise hierarchy	String	25	N
WAREHOUSE_KEY	Unique identifier for a warehouse	String	25	N
FISCAL_YEAR	Fiscal year ID	Integer	4	N
FISCAL_WEEK	Fiscal week ID	Integer	2	N
EOP_INVENTORY_UNITS	Describes the total DC EOP inventory units	Integer	22	Y
EOP_ON_ORDER_UNITS	Describes the total DC EOP on order units	Integer	22	Y

DC Allocation Specification (ASH_DC_ALLOCATION_TBL)

Table 2–23 Distribution Center Allocation Standard Interface Specification[1]

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
WAREHOUSE_KEY	Unique identifier for a warehouse	String	25	N
MERCHANDISE_KEY	Key for the item level in the merchandise hierarchy	String	25	N
MERCHANDISE_LEVEL	The level of the merchandise key. Must be at the level of sales optimization in order to be inserted into WAREHOUSE_INV_ TBL.	String	32	N
LOCATION_KEY	Key for the item level in the location hierarchy	String	25	N
FRACTION	Percentage of inventory, expressed as a value 0 - 1	Decimal	8,6	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.

MH Rename Specification (ASH_MHRENAME_TBL)

Table 2–24 Merchandise Hierarchy Rename Standard Interface Specification

Field Name	Field 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

LH Rename Specification (ASH_LHRENAME_TBL)

Table 2–25 Location Hierarchy Rename Standard Interface Specification

Field Name	Field 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

MH CDA Specification (ASH_MH_CDA_TBL)

Table 2–26 Merchandise Hierarchy CDA Standard Interface Specification[1]

Field Name	Field 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
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.

LH CDA Specification (ASH_LH_CDA_TBL)

Table 2–27 Location Hierarchy CDA Standard Interface Specification[1]

Field Name	Field 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
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.

Items CDA Specification (ASH_ITEMS_CDA_TBL)

Table 2–28 Items CDA Standard Interface Specification[1]

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_KEY	Unique identifier for merchandise hierarchy	String	25	N
LOCATION_KEY	Unique identifier for location hierarchy	String	25	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
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.

Business Rule Instances Specification (ASH_BRM_INSTANCE_TBL)

Table 2–29 Business Rule Instances Standard Interface Specification

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_KEY	Key for this level of the hierarchy	String	50	N
MERCHANDISE_LEVEL	ID for this level of the hierarchy	String	50	N
LOCATION_KEY	Key for this level of the hierarchy	String	50	N
LOCATION_LEVEL	ID for this level of the hierarchy	String	50	N
RULE_NAME	The name of the business rule associated with the item.	String	64	N
RULE_VALUE	The business rule value assigned to the item.	String Note: Values < 1 should be expressed as 0.n.	100	N
ATTRIB1_VALUE	The specific value associated with the item for custom attribute 1.	String	100	Y
ATTRIB2_VALUE	The specific value associated with the item for custom attribute 2.	String	100	Y
DELETE_FLAG	A flag to indicate whether the instance is to be deleted or inserted. 0 = insert (the default). 1 = delete.	Integer	1	Y

Demand Parameters Specification (ASH_PARAMETER_VALUES_TBL)

Table 2–30 Demand Parameters Standard Interface Specification

Field Name	Field 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
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

Price Ladders Specification (ASH_PRICE_LADDERS_TBL)

Table 2–31 Price Ladders Standard Interface Specification[1]

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
CLIENT_LADDER_ID	Unique identifer for the price ladder (i.e., unique per price ladder string)	Integer		N
MERCHANDISE_KEY	Key for this level of the merchandise hierarchy	String	25	N
MERCHANDISE_LEVEL	Description of this level of the merchandise hierarchy	String	50	N
LOCATION_KEY	Key for this level of the location hierarchy	String	25	N
LOCATION_LEVEL	Description of this level of the location hierarchy	String	50	N
PRICE_VALUE_TYPE	Percentage Off Original Retail Price (PO), Percentage Off Ticketed Price (PT), or Price Point (PP)	String	2	N
PRICE_LADDER_DESC	Price ladder name displayed in the UI.	String	50	Y
MODEL_FLAG	A model run indicator. R = price ladder used for optimization.	String	2	Y
ITEM_PRG_FLAG	Indicates if price ladder is used for the optimization of individual items only (ITEM), Pricing Groups only (PRG), or both (BOTH).	String	4	N

Table 2–31 (Cont.) Price Ladders Standard Interface Specification[1]

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
ACCOUNTING_TYPE	The accounting type can be either Permanent (PERM) or Temporary (TEMP).	String	4	N
PRICE_POINT	If price ladder type is PP, then this contains price point values. Note: Either PRICE_POINT or PRICE_PCT_OFF must be Not Null, depending on the value of the PRICE_VALUE_TYPE column.	Decimal	7,2	Y
PRICE_PCT_OFF	If price value type is PO, then this contains the percentage off (a value between 0 and 1). Note: Either PRICE_POINT or PRICE_PCT_OFF must be Not Null, depending on the value of the PRICE_VALUE_TYPE column.	Decimal	3,2	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.

Seasonalities Specification (ASH_SEASONALITY_MAPS_TBL and ASH_SEASONALITY_VALUES_TBL)

The seasonalites interface populates two tables in Price.

Table 2–32 Seasonalities (Maps) Standard Interface Specification

Field Name	Field 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
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 2–33 Seasonalities (Values) Standard Interface Specification[1]

Field Name	Field 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.

Cluster Specification (ASH_CLUSTER_LOAD_TBL)

Table 2–34 Cluster Standard Interface Specification

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
CLUSTER_SET_KEY	Unique identifier of the cluster set	String	25	Y
CLUSTER_SET_DESCRIPTION	Unique description of the cluster set	String	50	Y
CLUSTER_KEY	Unique identifier of the client	String	25	Y
CLUSTER_DESCRIPTION	Unique description of the cluster	String	50	Y
LOCATION_KEY	Unique identifier from the Location Hierarchy table	String	25	Y

Cluster Mapping Specification (ASH_CLUSTER_MAPPING_TBL)

Table 2–35 Cluster Mapping Standard Interface Specification

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
CLUSTER_SET_KEY	Unique identifier of the cluster set	String	25	Y
MERCHANDISE_KEY	Unique identifier from the Merchandise Hierarchy table	String	25	Y

CSH Levels Specification

Table 2–36 Cluster Levels Standard Interface Specification

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

3

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

Price provides two scripts that stage, transform, and load data into the target database tables in the Price 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.dat or cal.datafeed) to correspond to ASH_CAL_TBL. No specific file extension is required for the input files.

Table 3–1 Flat Files Names

Example File Name	File Content
mh.dat	Merchandise Hierarchy
lh.dat	Location Hierarchy
cal.dat	Calendar
items.dat	Items
sales.dat	Sales
budget.dat	Budget
mdtaken.dat	Markdowns Taken

Table 3–1 (Cont.) Flat Files Names

Example File Name	File Content
promo.dat	Promotions
dci.dat	Warehouse Inventory
dc_allocation.dat	Warehouse Allocation
mhrename.dat	MH Rename
lhrename.dat	LH Rename
mh_cda.dat	Merchandise Hierarchy Configurable Data Attributes
lh_cda.dat	Location Hierarchy Configurable Data Attributes
items_cda.dat	Item Configurable Data Attributes
brm_instance.dat	Business Rule Instances
parameter_values.dat	Demand Parameters
price_ladders.dat	Price ladders
seasonality_maps.dat	Seasonality Maps
seasonality_values.dat	Seasonality Values
clusters.dat	Clusters
merchclusters.dat	Cluster Mapping
clusterhierarchy.dat	Cluster Hierarchy
mvhistmd.dat	MVHistMarkdowns

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 ASH staging tables. The second script, **pl_load_data.sh**, loads the staged data into the Price 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 3–2 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
-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 3–3 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
-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 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 Price 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 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 Price 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 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 the Price UI.

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 the Price UI.

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 the Price UI.

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
the Price UI.

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 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 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 Price 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 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 Sales

Procedure: com.profitlogic.db.birch.LoadSales

Source Tables:

- ASH_SALES_TBL
- MERCHANDISE_TBL
- LOCATION_TBL
- ITEMS_TBL
- ASH_CP_TBL
- PERIODS_TBL

Target Table: ACTIVITIES

Description: The sales data is loaded from ASH_SALES_TBL, which is populated at the sales level for merchandise. This procedure loads data that is aggregated to the optimization level. It is stored in the ACTIVITIES table at the optimization level. Any number of weeks of data can be provided in ASH_SALES_TBL. The load procedure processes one week at a time, inserting new data or updating existing data.

Load Markdowns Taken

Procedure: com.profitlogic.db.birch.LoadMarkdownsTaken

Source Tables:

- ASH_MDTAKEN_TBL
- ITEMS_TBL
- MERCHANDISE_HIERARCHY_TBL
- LOCATION_HIERARCHY_TBL
- PERIODS_TBL

Target Tables:

- HIST_MARKDOWNS_ARCH_TBL
- HIST_MARKDOWNS_TBL

Description: This procedure compares the records from the current week's ASH_MDTAKEN_TBL feed with the records in HIST_MARKDOWNS_TBL and archives all the matches in HIST_MARKDOWNS_ARCH_TBL. It then replaces the records in HIST_MARKDOWNS_TBL with the new records from ASH_MDTAKEN_TBL.

Load Budget

Procedure: com.profitlogic.db.birch.LoadBudget

Source Tables:

- ASH_BUDGET_TBL
- ASH_CP_TBL
- MERCHANDISE_HIERARCHY_TBL
- LOCATION_HIERARCHY_TBL
- PERIODS_TBL

Target Table: P4P_BUDGET

Description: Budget data is provided on a monthly basis. (PERIODS_TBL.PERIOD_TYPE='FM' is used in the load query.) This procedure completely re-loads data into the P4P_BUDGET table from ASH_BUDGET_TBL, based on the worksheet level specified in ASH_CP_TBL.

Load Promotions

Procedure: com.profitlogic.db.birch.LoadPromotions

Source Tables:

- ASH_PROMO_TBL
- ITEMS_TBL
- MERCHANDISE_HIERARCHY_TBL
- LOCATION_HIERARCHY_TBL
- TCLOSE_TBL
- LTCLOSE_TBL

Target Tables:

- PLANNED_PROMOS_TBL
- PLANNED_PROMOS_MAPS_TBL

Description: This procedure completely re-loads the data in PLANNED_PROMOS_TBL and PLANNED_PROMOS_MAPS_TBL from ASH_PROMO_TBL.

Load Warehouses Table

Procedure: com.profitlogic.db.birch.LoadWarehouses

Source Table: ASH_DCI_TBL

Target Table: WAREHOUSES_TBL

Description: This procedure completely re-loads the data that describes warehouses (client key, type, description, and location) from ASH_DCI_TBL to WAREHOUSES_TBL.

Load Dc Inventory

Procedure: com.profitlogic.db.birch.LoadDcInventory

Source Tables:

- ASH_DCI_TBL

- WAREHOUSES_TBL
- ASH_CP_TBL
- ASH_MHL_TBL
- MERCHANDISE_TBL
- PERIODS_TBL

Target Table: WAREHOUSE_INV_TBL

Description: This procedure inserts new inventory data or updates existing data at the optimization level into WAREHOUSE_TBL from ASH_DCI_TBL. The data in ASH_DCI_TBL is at the merchandise level that corresponds to INTERSECT_NAME = SALES in ASH_CP_TBL.

Load Warehouse Allocation

Procedure: com.profitlogic.db.birch.LoadWarehouseAllocation

Source Tables:

- ASH_DC_ALLOCATION_TBL
- WAREHOUSES_TBL
- ASH_CP_TBL
- MERCHANDISE_HIERARCHY_TBL
- LOCATION_HIERARCHY_TBL

Target Table: WAREHOUSE_ALLOCATION_TBL

Description: This procedure inserts new data or updates existing data at the optimization level from ASH_DC_ALLOCATION_TBL into WAREHOUSE_ALLOCATION_TBL, which contains data about what fraction of a particular item's inventory is stored in which warehouse.

Load Business Rule Instances

Procedure: com.profitlogic.db.birch.LoadBRInstances

Source Tables:

- ASH_BRM_INSTANCE_TBL
- BRM_VALUE_DEFINITIONS_TBL
- BRM_KEY_LEVELS_TBL
- MERCHANDISE_HIERARCHY_TBL
- LOCATION_HIERARCHY_TBL

Target Tables:

- BRM_INSTANCE_TBL
- BRM_INSTANCE_CHANGE_TBL

Description: This procedure re-loads business rule instances from ASH_BRM_INSTANCE_TBL to BRM_INSTANCE_TBL. If existing data is modified or deleted, the old data is written to BRM_INSTANCE_CHANGE_TBL to preserve the change history.

Load Sendbacks

Procedure: com.profitlogic.db.birch.LoadMarkdownsSendback

Source Table: PL_MARKDOWN_SENDBACK view

Target Tables:

- HIST_MARKDOWNS_ARCH_TBL
- HIST_MARKDOWNS_TBL

Description: This procedure compares the records in HIST_MARKDOWNS_TBL with the records in the current week's PL_MARKDOWN_SENDBACK view and archives the matches in HIST_MARKDOWNS_ARCH_TBL. It then replaces the records in HIST_MARKDOWNS_TBL with records from the PL_MARKDOWN_SENDBACK view that match on ITEM_ID and CALENDAR_DT/EFFECTIVE_DT. That is, all the latest information on all item markdowns recorded by Price can be found in HIST_MARKDOWNS_TBL; all information on prior markdowns recorded by Price can be found in HIST_MARKDOWNS_ARCH_TBL.

Load Model Start Date

Procedure: com.profitlogic.db.cdw.LoadModelStartDate

The four possible model start options in the IR_MODEL_START_OPTION view are:

- InventoryRatio - the first date when (inventory/(cumulative_sales_to_date + inventory + on_order)) > a defined threshold
- StoreRatio - the first date when ((stores_with_inventory)/(stores_in_region)) > a defined threshold
- PlannedStart - the date of the first sale after the planned start date (a business rule value)
- Custom - a value defined in the IR_MODEL_START view

Source Tables:

InventoryRatio option	ACTIVITIES
StoreRatio option	ACTIVITIES ITEM_ACTUALS_MV LTCLOSE_TBL
PlannedStart option	ACTIVITIES FIRST_SALE_DT_MV ITEMS view getBRValue function (which retrieves data from BRM_INSTANCE_TBL, TCLOSE_TBL, and LTCLOSE_TBL)
Custom option	IR_MODEL_START view

Target Table: ITEMS_TBL

Description: This procedure updates MODEL_START_DT in ITEMS_TBL, based on the model start date option specified in the IR_MODEL_START_OPTION view.

The model start date is the first date that an item is considered to be available for sale. It is always represented as the Sunday preceding the actual specified date.

Load Materialized Views

Procedure: com.profitlogic.db.cdw.LoadMaterializedViews

Source Tables:

- ITEMS_TBL

- PERIODS_TBL
- ACTIVITIES

Target Tables:

- DB_LAST_ACTUAL_MV
- ITEM_HIST_MARKDOWNS_MV
- ITEM_METRICS_TBL

Description: This procedure refreshes DB_LAST_ACTUAL_MV and ITEM_HIST_MARKDOWNS_MV.

It re-loads ITEM_METRICS_TBL from the ACTIVITIES data. If the FIRST_RECEIPT_DT is specified in ITEMS_TBL, then only the data received after that date is loaded. If the FIRST_RECEIPT_DT is not specified, then all the data is loaded.

The ITEMS_METRICS_TBL is the source for the following views:

- FIRST_INVENTORY_DT_MV
- FIRST_SALE_DT_MV
- ITEM_ACTUALS_MV
- ITEM_AVG_COST_MV
- LAST_TICKET_PRICE_DT_MV

Load Scenarios

Procedure: com.profitlogic.db.birch.LoadScenarios

Source Table: ACTIVITIES

Target Table: SCENARIOS_TBL

Description: This procedure deletes and then inserts data from ACTIVITIES into SCENARIOS_TBL, where Scenario_ID = 1.

Load Internal Promos

Procedure: com.profitlogic.db.birch.LoadInternalPromos

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 Price Ladders

Procedure: com.profitlogic.db.birch.LoadPriceLadders

Source Table: ASH_PRICE_LADDERS_TBL

Target Tables:

- PRICE_LADDERS_TBL
- PRICE_LADDER_VALUES_TBL

Description: This procedure truncates and reloads PRICE_LADDERS_TBL and PRICE_LADDER_VALUES_TBL. It populates client_search_levels_tbl with "PRICE_LADDERS" search name and rank ordered sequence base on location and merchandise level sequences.

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 Collections Sendback

Procedure: com.profitlogic.db.birch.LoadCollectionsSendback

Target Tables:

- ITEM_DATA_TBL
- COLLECTIONS_TBL
- COLLECTION_MAPS_TBL

Description: Any changes or additions to pricing groups that users implement via the Price UI override the assignment of items to pricing groups that are made by the LoadCollectionsAuto procedure and are processed by the LoadCollectionsSendback procedure.

Load Collections Auto

Procedure: com.profitlogic.db.birch.LoadCollectionsAuto

Source Tables:

- all items from IR_ITEM_COLLECTION except those that have already been auto-collected

Target Tables:

- COLLECTIONS_TBL
- COLLECTION_MAPS_TBL

Description: This procedure determines how to group items into pricing groups.

Load Clusters

Procedure: com.profitlogic.db.birch.LoadClusters

Source Tables:

- ASH_CLUSTER_LOAD_TBL

- LOCATION_HIERARCHY_TBL
- CLIENT_HIERARCHY_LEVELS_TBL

Target Tables:

- CDW_LOCATION_CLUSTER_XREF_TBL
- CDW_CLUSTER_SET_TBL
- CDW_CLUSTER_TBL

Description: This procedure loads cluster information, cluster set information, and maps location to clusters.

Load Merchandise Cluster

Procedure: com.profitlogic.db.birch.LoadMerchCluster

Source Tables:

- ASH_CLUSTER_MAPPING_TBL
- ASH_CP_TBL
- MERCHANDISE_HIERARCHY_TBL
- CDW_CLUSTER_SET_TBL

Target Tables: CDW_MERCH_CLUSTER_XREF_TBL

Description: This procedure maps merchandise IDs to cluster sets.

Load Cluster Hierarchy

Procedure: com.profitlogic.db.birch.LoadClusterHierarchy

Source Tables:

- ASH_CLUSTER_LOAD_TBL
- CDW_CLUSTER_SET_TBL
- CDW_CLUSTER_TBL
- LOCATION_HIERARCHY_TBL
- CLIENT_HIERARCHY_LEVELS_TBL
- CDW_LOC_HIERARCHY_ORIG_TBL

Target Tables:

- LOCATION_HIERARCHY_TBL
- CDW_LOCATION_CLUSTER_XREF_TBL

Description: This procedure loads the cluster structure (CHAIN > CLUSTER_SET > CLUSTER > OPTIMIZATION LOCATION) into LOCATION_HIERARCHY_TBL. This replaces the original Location Hierarchy information. The procedure also maps location to clusters.

Load MaterializedViewsHistMarkdowns

Procedure: com.profitlogic.db.birch.LoadMaterializedViewsHistMarkdowns

Description: This procedure loads the ITEM_HIST_MARKDOWNS_MV table. This table contains the count of taken markdowns per item after the model start date. This

table is not used by the default product configuration but may be used in a custom configuration.

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 Dependency Tree" on page 30).

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

Table 3–4 Failed Records Table Names

Staging Table	Failed Record Table
ASH_SALES_TBL	ASH_SALES_TBL_BAD
ASH_PROMO_TBL	ASH_PROMO_TBL_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 3–5 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 Price. In order to customize the threshold values, you must create a properties file and load it into Price.

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 Price 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 3-15.

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 3–8, " 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 Price 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 3–6 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 3–7 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
LoadItems	ASH_ITEMS_TBL	Success	0
LoadCalendars	ASH_CAL_TBL	Success	0
LoadDcInventory	ASH_DCI_TBL	Success	0
LoadLocationHierarchy	ASH_LH_CDA_TBL	Success	0
LoadMerchandiseHierarchy	ASH_MH_CDA_TBL	Success	0
LoadPromotions	ASH_PROMO_TBL	Success	0
LoadSales	ASH_SALES_TBL	Success	0
LoadBudget	ASH_BUDGET_TBL	Success	0
No transformation of data	ASH_CP_TBL	Success	0
LoadLHKeyRename	ASH_LHRENAME_TBL	Success	0
LoadMHKeyRename	ASH_MHRENAME_TBL	Success	0

Table 3–7 (Cont.) Sample Standard Load Data Validation Report

LOAD_PROCEDURE	SOURCE	DATA_ VALIDATION_ STATUS	NUM_ BAD_ RECORDS
LoadMDTaken	ASH_MDTAKEN_TBL	Success	20
LoadItems	ASH_ITEMS_CDA_TBL	Success	0
LoadWarehouseAllocation	ASH_DC_ALLOCATION_TBL	Failure	50
LoadBRInstances	ASH_BRM_INSTANCE_TBL	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 3–8 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.
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.

Table 3–8 (Cont.) Standard Load Error Messages

Number	Error Message
105	The sequence for the CHAIN level should be defined as 1 in the levels table.
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.
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.
Load Markdowns Taken Errors	

Table 3–8 (Cont.) Standard Load Error Messages

Number	Error Message
1100	In the markdowns taken table (ASH_MDTAKEN_TBL), if price value type is 'PO' or 'PT', then the price percent off value has to be a non-negative fraction and the price point value has to be null. If the price value type is 'PP', then the price point value has to be a non-negative number and the price percent off value has to be null.
1101	In the markdowns taken table (ASH_MDTAKEN_TBL), only one markdown can be loaded for a unique combination of merchandise, location, and effective date.
1102	In the markdowns taken table (ASH_MDTAKEN_TBL), at least one merchandise key has a value of NULL. A merchandise key cannot have a value of NULL.
1103	In the markdowns taken table (ASH_MDTAKEN_TBL), at least one location key has a value of NULL. A location key cannot have a value of NULL.
1104	In the markdowns taken table (ASH_MDTAKEN_TBL), at least one effective date has a value of NULL. An effective date cannot have a value of NULL.
1105	In the markdowns taken table (ASH_MDTAKEN_TBL), at least one accounting type is either null or has a value that is not allowed. Acceptable values are TEMP or PERM.
1106	In the markdowns taken table (ASH_MDTAKEN_TBL), at least one price value type is either null or has a value that is not allowed. Acceptable values are PT, PO, or PP.
1107	The MERCHANDISE_KEY in the markdowns taken table (ASH_ MDTAKEN_TBL) is not at the optimization level.
1108	The LOCATION_KEY in the markdowns taken table (ASH_ MDTAKEN_TBL) is not at the optimization level.
Load Promotions Errors	
1200	A promotion with an end date occurring prior to the last date in the sales history cannot be loaded into the promotions table (ASH_ PROMO_TBL).
1201	A promotion in the promotions table (ASH_PROMO_TBL) cannot be defined in terms of both a specific price point and a percentage off.
1204	A promotion in the promotions table (ASH_PROMO_TBL) cannot be defined as a negative percent off or be a value > 1.
1205	The promotion flag in the promotions table (ASH_PROMO_TBL) can have a value of either 1 or -1.
1206	In the promotions table (ASH_PROMO_TBL), at least one merchandise key has a value of NULL. A merchandise key cannot have a value of NULL.
1207	In the promotions table (ASH_PROMO_TBL), at least one merchandise level has a value of NULL. A merchandise level cannot have a value of NULL.
1208	In the promotions table (ASH_PROMO_TBL), at least one location key has a value of NULL. A location key cannot have a value of NULL.
1209	In the promotions table (ASH_PROMO_TBL), at least one location level has a value of NULL. A location level cannot have a value of NULL.

Table 3–8 (Cont.) Standard Load Error Messages

Number	Error Message
1210	In the promotions table (ASH_PROMO_TBL), at least one promotion start date has a value of NULL. A promotion start date cannot have a value of NULL.
1211	In the promotions table (ASH_PROMO_TBL), at least one promotion end date has a value of NULL. A promotion end date cannot have a value of NULL.
1212	In the promotions table (ASH_PROMO_TBL), at least one promotion type has a value of NULL. A promotion type cannot have a value of NULL.
1213	A record in the promotions table (ASH_PROMO_TBL) contains merchandise that is not found in the merchandise hierarchy.
1214	A record in the promotions table (ASH_PROMO_TBL) contains a location that is not found in the location hierarchy.
Load Items Errors	
1300	More than one record in the items table (ASH_ITEMS_TBL) has the same combination of merchandise and location.
1301	A record in the items table (ASH_ITEMS_TBL) contains merchandise that is not found in the merchandise hierarchy at the optimization level.
1302	A record in the items table (ASH_ITEMS_TBL) contains a location that is not found in the location hierarchy at the optimization level.
1303	In the items table (ASH_ITEMS_TBL), the number of processed rows does not match the original number of input rows.
1304	In the items table (ASH_ITEMS_TBL), the loading of the locations did not complete.
1305	In the items table (ASH_ITEMS_TBL), at least one merchandise key has a value of NULL. A merchandise key cannot have a value of NULL.
1306	In the items table (ASH_ITEMS_TBL), at least one location key has a value of NULL. A location key cannot have a value of NULL.
1307	In the items table (ASH_ITEMS_TBL), at least one unit cost has NULL value or negative value. A unit cost can only be a positive number.
1308	In the items CDA table (ASH_ITEMS_CDA_TBL), at least one merchandise key has a value of NULL. A merchandise key cannot have a value of NULL.
1309	In the items CDA table (ASH_ITEMS_CDA_TBL), at least one location key has a value of NULL. A location key cannot have a value of NULL.
1310	A record in the items CDA table (ASH_ITEMS_CDA_TBL) contains merchandise that is not found in the merchandise hierarchy on the optimization level.
1311	A record in the items CDA table (ASH_ITEMS_CDA_TBL) contains a location that is not found in the location hierarchy on the optimization level.
1312	More than one record in the items CDA table (ASH_ITEMS_CDA_ TBL) has the same combination of merchandise and location.
Load DC Inventory Errors	

Table 3–8 (Cont.) Standard Load Error Messages

Number	Error Message
1400	In the DC inventory table (ASH_DCI_TBL), at least one merchandise key has a value of NULL. A merchandise key cannot have a value of NULL.
1401	In the DC inventory table (ASH_DCI_TBL), at least one warehouse key has a value of NULL. A warehouse key cannot have a value of NULL.
1402	In the DC inventory table (ASH_DCI_TBL), at least one fiscal year has a value of NULL. A fiscal year cannot have a value of NULL.
1403	In the DC inventory table (ASH_DCI_TBL), at least one fiscal week has a value of NULL. A fiscal week cannot have a value of NULL.
1404	Cannot determine OPTIMIZATION merchandise level. Check your configuration.
1405	The MERCHANDISE_KEY in the DC inventory table (ASH_DCI_TBL) is not at the sales level.
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.

Table 3–8 (Cont.) Standard Load Error Messages

Number	Error Message
1605	The new location key in the location hierarchy rename table (ASH_LHRENAME_TBL) is already present in the location hierarchy.
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.
2604	The new merchandise key in the merchandise hierarchy rename table (ASH_MHRENAME_TBL) contains duplicate values.

Table 3–8 (Cont.) Standard Load Error Messages

Number	Error Message
2605	The new merchandise key in the merchandise hierarchy rename table (ASH_MHRENAME_TBL) is already present in the merchandise hierarchy.
Worksheet Errors	
3000	Inference rule IR_WORKSHEET_IDS is not configured correctly and returns more than one row for the same item.
3001	A worksheet contains duplicate combinations of hierarchies 1, 2, 3, and 4.
3002	A worksheet contains duplicate combinations of merchandise level and location level.
Load Budget Errors	
3501	The budget table (ASH_BUDGET_TBL) is empty.
3522	The composite key (MERCHANDISE_LEVEL, LOCATION_LEVEL, FISCAL_YEAR, and FISCAL_WEEK) in the budget table (ASH_BUDGET_TBL) contains null values.
3523	The composite key (MERCHANDISE_LEVEL, LOCATION_LEVEL, FISCAL_YEAR, and FISCAL_WEEK) in the budget table (ASH_BUDGET_TBL) is not unique.
3524	In the budget table (ASH_BUDGET_TBL), at least one location key has a value of NULL. A location key cannot have a value of NULL.
3525	In the budget table (ASH_BUDGET_TBL), at least one fiscal year has a value of NULL. A fiscal year cannot have a value of NULL.
3526	In the budget table (ASH_BUDGET_TBL), at least one fiscal month has a value of NULL. A fiscal month cannot have a value of NULL.
3531	PERIODS_TBL is empty.
3551	LOCATION_HIERARCHY_TBL is empty.
3552	Merchandise found that does not exist in MERCHANDISE_HIERARCHY_TBL at the required level (Worksheet).
3553	Location found that does not exist in LOCATION_HIERARCHY_TBL at the required level (Worksheet).
Load DC Allocation Errors	
3601	The LOCATION_KEY in the distribution center allocation table (DC_ALLOCATION_TBL) is not at the optimization level.
3603	The value in WAREHOUSE_KEY in the distribution center allocation table (DC_ALLOCATION_TBL) does not exist in WAREHOUSE_TBL.
3604	The value in MERCHANDISE_KEY in the distribution center allocation table (DC_ALLOCATION_TBL) does not exist in MERCHANDISE_HIERARCHY_TBL.
3605	In the DC allocation table (ASH_DC_ALLOCATION_TBL), at least one warehouse key has a value of NULL. A warehouse key cannot have a value of NULL.
3606	In the DC allocation table (ASH_DC_ALLOCATION_TBL), at least one merchandise key has a value of NULL. A merchandise key cannot have a value of NULL.

Table 3–8 (Cont.) Standard Load Error Messages

Number	Error Message
3607	In the DC allocation table (ASH_DC_ALLOCATION_TBL), at least one location key has a value of NULL. A location key cannot have a value of NULL.
3608	In the DC allocation table (ASH_DC_ALLOCATION_TBL), at least one fraction has NULL value or negative value. A fraction can only be a positive number.
3609	The value in MERCHANDISE_LEVEL in the distribution center allocation table (ASH_DC_ALLOCATION_TBL) does not exist in CLIENT_HIERARCHY_LEVELS_TBL.
Load Sales Errors	
3701	NET_SALES_UNITS in the sales table (ASH_SALES_TBL) is mandatory and cannot have a value of null.
3702	NET_SALES_AMT in the sales table (ASH_SALES_TBL) is mandatory and cannot have a value of null.
3703	GROSS_SALES_UNITS in the sales table (ASH_SALES_TBL) is mandatory and cannot have a value of null.
3704	GROSS_SALES_AMT in the sales table (ASH_SALES_TBL) is mandatory and cannot have a value of null.
3705	EOP_INVENTORY_UNITS in the sales table (ASH_SALES_TBL) is mandatory and cannot have a value of null.
3706	EOP_ON_ORDER_UNITS in the sales table (ASH_SALES_TBL) is mandatory and cannot have a value of null.
3707	EOP_STORE_NUM_WITH_INV in the sales table (ASH_SALES_ TBL) is mandatory and cannot have a value of null.
3708	EOP_STORE_NUM_WITH_OO in the sales table (ASH_SALES_TBL) is mandatory and cannot have a value of null.
3709	CURRENT_RETAIL in the sales table (ASH_SALES_TBL) is mandatory and cannot have a value of null.
3710	CURRENT_INV_PRICE in the sales table (ASH_SALES_TBL) is mandatory and cannot have a value of null.
3711	The resent data does not match the data from the previous week.
3712	More than one week of data in the interface has already been processed.
3713	A record in the sales table (ASH_SALES_TBL) contains merchandise that is not found in the merchandise hierarchy.
3714	A record in the sales table (ASH_SALES_TBL) contains a location that is not found in the location hierarchy.
3715	In the sales table (ASH_SALES_TBL), at least one merchandise key has a value of NULL. A merchandise key cannot have a value of NULL.
3716	In the sales table (ASH_SALES_TBL), at least one location key has a value of NULL. A location key cannot have a value of NULL.
3717	In the sales table (ASH_SALES_TBL), at least one fiscal year has a value of NULL. A fiscal year cannot have a value of NULL.
3718	In the sales table (ASH_SALES_TBL), at least one fiscal week has a value of NULL. A fiscal week cannot have a value of NULL.

Table 3–8 (Cont.) Standard Load Error Messages

Number	Error Message
3719	In the sales tables (ASH_SALES_TBL), at least one combination of merchandise_key and location_key is not found in ITEMS_TBL.
3720	The active sales weeks (ActiveSalesWeeks) specified for data archiving is invalid. The value must be a positive integer or zero.
3721	The ActiveSalesWeeks entry is not found in the P4P_PARAMS table.
3722	The ACTIVITIES table is empty.
3723	An error occurred getting a sequence for the SALES level.
3724	An error occurred getting a sequence for the OPTIMIZATION level.
Load BRM Rules Errors	
3801	The BRM_RULE_DEFINITION_TBL is empty and needs to be populated.
Load Flexible Store Clustering Errors	
3901	Cluster Definition Load has an invalid Location LOCATION_CLIENT_ID. It should match what is defined in CDW_LOC_HIERARCHY_ORIG_TBL <client_load_id>.
3902	The Cluster Mapping Load has an invalid merchandise MERCHANDISE_CLIENT_ID. It should match what is defined in MERCHANDISE_HIERARCHY_TBL <client_load_id>.
3903	The Cluster Mapping Load has an invalid cluster set CLUSTER_SET_CLIENT_ID. It should match what is defined in CDW_CLUSTER_SET_TBL <cluster_client_id>.
3904	Cluster to Location mapping warning: The following cluster set does not contain every location.
3905	Cluster to Location mapping warning: The following location is not contained in every cluster.
3906	In the cluster hierarchy levels table (ASH_CSHL_TBL), at least one cluster level has a value of NULL. A cluster level cannot have a value of NULL.
3907	In the cluster hierarchy levels table (ASH_CSHL_TBL), at least one level sequence level has a value of NULL. A level sequence cannot have a value of NULL.
3908	In the cluster hierarchy levels table (ASH_CSHL_TBL), the entries in LEVEL_SQC are not sequential.
3909	The cluster hierarchy levels table (ASH_CSHL_TBL) should have level sequences starting with 1.
3910	In the cluster hierarchy levels table (ASH_CSHL_TBL), CHAIN is not assigned a sequence value (LEVEL_SQC) of 1.
3911	The cluster hierarchy levels table (ASH_CSHL_TBL) should have a level ending with 4.
3912	The item location to merchandise mapping in ASH_ITEM_TBL does not match the cluster mapping defined in the CDW_MERCH_CLUSTER_XREF_TBL.
3913	The ASH_CLUSTER_LOAD_TBL cluster set has a duplicate location defined within it <Cluster Set Client ID>.
3914	The ASH_CLUSTER_LOAD_TBL cluster_set_client_id is NULL.
3915	The ASH_CLUSTER_LOAD_TBL cluster_client_id is NULL.

Table 3–8 (Cont.) Standard Load Error Messages

Number	Error Message
3916	The ASH_CLUSTER_LOAD_TBL location_client_id is NULL.
3917	The ASH_CLUSTER_MAPPING_TBL cluster_set_client_id is NULL
3918	The ASH_CLUSTER_MAPPING_TBL merchandise_client_id is NULL.
3919	There is no mapping for clusters in the ASH_CP_TBL.
3920	The LOCATION_LEVEL in ASH_CP_TBL for CLUSTER is invalid.
3921	The MERCHANDISE_LEVEL in ASH_CP_TBL for CLUSTER is invalid.
3922	The following merchandise does not have a cluster mapped to it <merch_client_load_id>.
3923	The CDW_CLUSTER_SET_TBL is empty.
3924	The CDW_CLUSTER_TBL is empty.
3925	There are no level descriptions for clusters in the CLIENT_ HIERARCHY_LEVELS_TBL. There must be LOCATION and either CLUSTER or CUST_LOCATION entries to load cluster information.
3926	There are no level descriptions for clusters in the CLIENT_ HIERARCHY_LEVELS_TBL. There must be both LOCATION and CUST_LOCATION entries to load cluster information.
Load To Date Metrics	
4001	The activities information table (ACTIVITIES) is empty. Run the LoadSales procedure to load activities and then re-run LoadToDateMetrics.
Load BR Instances Errors	
4100	A business rule cannot have more than one value definition (BRM_ VALUE_DEFINITIONS_TBL) defined. Multi-valued business rules are not supported.
4101	A business rule key (RULE_NAME, MERCHANDISE_LEVEL, LOCATION_LEVEL, ATTRIB1_VALUE, ATTRIB2_VALUE) in the business rules staging table (ASH_BRM_INSTANCE_TBL) is not legal.
4102	A business rule value (RULE_VALUE) in the business rules staging table (ASH_BRM_INSTANCE_TBL) is not in the permissible range.
4103	A business rule value (RULE_VALUE) in the business rules staging table (ASH_BRM_INSTANCE_TBL) is not in the permissible enumeration.
4104	No business rule definitions exist in table (BRM_RULE_ DEFINITION_TBL).
4105	In the business rule staging table (ASH_BRM_INSTANCE_TBL), at least one merchandise key has a value of NULL. A merchandise key cannot have a value of NULL.
4106	In the business rule staging table (ASH_BRM_INSTANCE_TBL), at least one merchandise level has a value of NULL. A merchandise level cannot have a value of NULL.
4107	In the business rule staging table (ASH_BRM_INSTANCE_TBL), at least one location key has a value of NULL. A location key cannot have a value of NULL.

Table 3–8 (Cont.) Standard Load Error Messages

Number	Error Message
4108	In the business rule staging table (ASH_BRM_INSTANCE_TBL), at least one location level has a value of NULL. A location level cannot have a value of NULL.
4109	In the business rule staging table (ASH_BRM_INSTANCE_TBL), at least one rule name has a value of NULL. A rule name cannot have a value of NULL.
4110	A record in the business rule staging table (ASH_BRM_INSTANCE_TBL) contains merchandise that is not found in the merchandise hierarchy.
4111	A record in the business rule staging table (ASH_BRM_INSTANCE_TBL) contains a location that is not found in the location hierarchy.
4112	A record in the business rule staging table (ASH_BRM_INSTANCE_TBL) contains merchandise that is not found in the merchandise hierarchy.
Partitioning Item_Data Conditions	
5010	ITEM_DATA view does not exist or could not be dropped.
5011	ITEM_DATA partition could not be dropped.
5012	ITEM_DATA table is not partitioned or does not exist.
5013	An index on the partitioned ITEM_DATA table could not be created.
5014	The ITEM_DATA table is already partitioned.
5015	The ITEM_DATA table is already non-partitioned.
Partitioning Item_Data Errors	
5100	Invalid input number for number of ITEM_DATA partitions.
5101	The ITEM_DATA table is empty.
5102	The ITEM_DATA table could not be distributed across the given number of partitions.
5103	The ITEM_DATA backup table is missing.
5104	The worksheet was not assigned a partition.
5105	The ITEM_DATA table's dependents were not all re-created.
5106	An error other than a missing base object error occurred when re-creating a dependent object.
5107	An error occurred dropping dependent other than nested dependencies.
Partitioning from LoadSWID	
5108	The ITEM_DATA table is not partitioned or does not exist.
5109	The high value of the ITEM_DATA partition key is not MAXVALUE.
5110	The ITEM_DATA table was not reset to one MAXVALUE partition.
5111	No submittal worksheets were found.
5112	The MAXVALUE partition in the ITEM_DATA table could not be dropped.
5113	The MAXVALUE partition could not be added to the ITEM_DATA table.
Load MHTbl Errors	

Table 3–8 (Cont.) Standard Load Error Messages

Number	Error Message
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).
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.
Load Price Ladders Errors	
8201	The merchandise key in ASH_PRICE_LADDERS_TBL cannot be null.
8202	The merchandise level in ASH_PRICE_LADDERS_TBL cannot be null.
8203	The location key in ASH_PRICE_LADDERS_TBL cannot be null.
8204	The location level in ASH_PRICE_LADDERS_TBL cannot be null.
8205	The price ladder ID in ASH_PRICE_LADDERS_TBL cannot be null.
8206	Merchandise found in ASH_PRICE_LADDERS_TBL that does not exist in MERCHANDISE_HIERARCHY_TBL.
8207	Location found in ASH_PRICE_LADDERS_TBL that does not exist in LOCATION_HIERARCHY_TBL.
8208	The found price ladder type is not in PT, PO, or PP.
8209	The model flag cannot be NULL.
8210	Accounting type is either NULL or has a value that is not allowed. Acceptable values are TEMP or PERM.
8211	If the price value type is PO or PT, then the price percent value must be a non-negative fraction and the price point value must be null. If the price value type is PP, then the price point value must be a non-negative number and the price percent off value must be null.
Load Seasonalities Error Messages	
8301	The merchandise key in ASH_SEASONALITY_MAPS_TBL cannot be null.

Table 3–8 (Cont.) Standard Load Error Messages

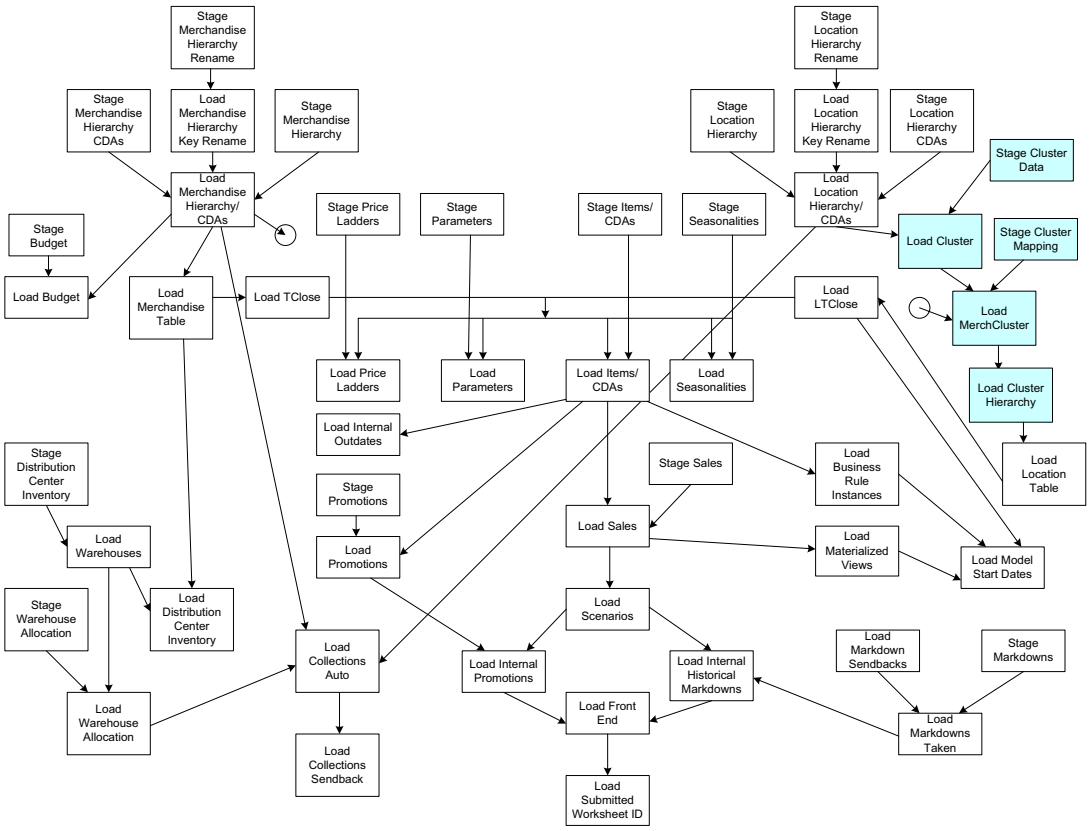
Number	Error Message
8302	The merchandise level in ASH_SEASONALITY_MAPS_TBL cannot be null.
8303	The location key in ASH_SEASONALITY_MAPS_TBL cannot be null.
8304	The location level in ASH_SEASONALITY_MAPS_TBL cannot be null.
8305	Merchandise found in ASH_SEASONALITY_MAPS_TBL that does not exist in MERCHANDISE_HIERARCHY_TBL.
8306	Location found in ASH_SEASONALITY_MAPS_TBL that does not exist in LOCATION_HIERARCHY_TBL.
8307	A NULL priority was found.
8308	A NULL seasonality ID in maps was found.
8309	A NULL seasonality ID in values was found.
8310	A NULL calendar date was found.
Data Archiving Errors	
11001	Records having a null RUN_ID have been found in the ITEM_DATA table.

Standard Load Dependency Tree

The load script loads data in the order specified in Figure 3–1, "Standard Load Dependency Tree." The load procedures that are shaded are only used with Flexible Store Clustering.

Figure 3–1 Standard Load Dependency Tree

Figure 3-1

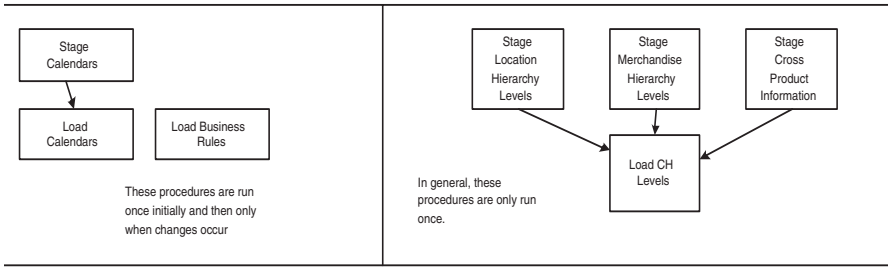


In addition, there are load procedures that are only run once

- Stage and load Calendars
- Load Business Rules

and load procedures that are run once initially and then only if the data changes:

- Stage LH Levels, MH Levels, and Cross Product Information
- Load CH Levels
- Load Cluster Levels (optional; only if using Flexible Clustering)



You can use the dependency tree to determine how to schedule the standard load (for example, when using third-party software). Each box in the dependency tree represents a single procedure. The load script calls the load procedures using the following names:

Table 3–1 Dependency Tree Procedures

Dependency Tree Procedure	Procedure Name
Load Calendars	LoadCalendars
Load CH Levels	LoadCHLevels
Load Merchandise Hierarchy Key Rename	LoadMHKeyRename
Load Merchandise Hierarchy (includes Merchandise Hierarchy CDAs)	LoadMerchandiseHierarchy
Load Merchandise Hierarchy Table	LoadMHTbl
Load TClose	LoadTCLOSE
Load Location Hierarchy Key Rename	LoadLHKeyRename
Load Location Hierarchy (includes Location Hierarchy CDAs)	LoadLocationHierarchy
Load Location Hierarchy Table	LoadLHTbl
Load LT Close	LoadLTCLOSE
Load Budget	LoadBudget
Load Items (includes Item CDAs)	LoadItems
Load Internal Outdates	LoadInternalOD
Load Promotions	LoadPromotions
Load Sales	LoadSales
Load Materialized Views	LoadMaterializedViews
Load Model Start Dates	LoadModelStartDate
Load Scenarios	LoadScenarios
Load Markdown Sendbacks	LoadMarkdownSendback
Load Markdowns Taken	LoadMarkdownsTaken
Load Internal Historical Markdowns	LoadInternalHM
Load Internal Promotions	LoadInternalPromo
Load Front End	LoadFrontEnd
Load Submitted Worksheet ID	LoadSWID
Load Warehouses	LoadWarehouses
Load Distribution Center Inventory	LoadDcInventory
Load Warehouse Allocation	LoadWarehouseAllocation
Load Business Rule Instances	LoadBRInstances
Load Parameters	LoadParameters
Load Price Ladders	LoadPriceLadders
Load Seasonalities	LoadSeasonalities
Load Auto-Collections	LoadCollectionsAuto

Table 3–1 (Cont.) Dependency Tree Procedures

Dependency Tree Procedure	Procedure Name
Load Collections Sendback	LoadCollectionsSendback
Load Cluster Mapping	LoadClusterMerch
Load Cluster Descriptions	LoadClusters
Load Cluster Hierarchy Structure	LoadClusterHierarchy

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 Price 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 defines the purpose or feature for the data, and is either OPTIMIZATION, SALES, WORKSHEET, CLUSTER, or DEFAULT LEVEL. Use the value CLUSTER to enable Flexible Clustering. For more information on Flexible Clustering, see the Price Configuration Guide.
- For each Key, identify the defining level of the merchandise hierarchy and location hierarchy.
- The cross products information is generally loaded only once.
- Sales cannot be loaded until optimization level and the sales level are defined. Worksheets must be defined before an optimization run can occur.

Cross Products Information Specification

Table 3–2 Cross Products Information Standard Interface Specification

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
INTERSECT_NAME	The key name (OPTIMIZATION, SALES, WORKSHEET, CLUSTER, or DEFAULT LEVEL)	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

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 3–3 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 3–4 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

Cluster Levels Standard Interface (ASH_CSHL_TBL)

The cluster levels interface is used to specify the names of a retailer's cluster 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 cluster levels information is generally loaded only once.

CSH Levels Specification

Table 3–5 Cluster Levels Standard Interface Specification

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

Standard Dataset

The Price standard dataset is a set of raw data provided as flat files with Price that

- is shipped with Price and is copied into the installation directory
- requires the front-end configuration grids in the installation directory
- contains five weeks of data
- contains only valid data, so no validation errors should occur during the standard load
- provides data that is sufficient to verify the installation and initial basic configuration of the product
- does not provide a complete set of data to explore the total functionality of Price
- provides data that can be loaded using the standard load procedures, and, once loaded, can be used to perform a model run

- provides data sufficient to permit the launching of the Price application and invoke the Price UI without any additional configuration after the model run is complete
- does not provide any error conditions
- provides one eligibility query for all the merchandise in the dataset. A subset of the merchandise can be run using the **PLBatch** command.
- includes the expected results of the optimization run
- supports markdowns and forecasts
- uses the default error threshold settings for data validation (procedures = 0 and specific validations = 100). This permits the dataset to fail on any validation error. All invalidated rows are store in appropriate "BAD" staged table.
- requires that an empty schema be created before the data is loaded (part of the standard installation)
- is staged and loaded using pl_stage_client.sh and pl_load_client.sh. See the beginning of this chapter for details about staging and loading data.
- is validated using published load validations
- imports the attributes such as promotions, markdowns, and business rules that are hierarchical in application of the attributes
- contains no sendback data. To test sendbacks, markdowns must be taken, which are then processed during the next weekly load.

Dataset Data

The data in the dataset consists of 3,107 items and 156 pricing groups and 4,682 items as part of pricing groups. Details about the data characteristics are shown in the following table. For information about the standard load procedure and the target table for each interface, see the beginning of this chapter.

Table 3–6 Price Dataset Data

Data Interface Table	Data Characteristics
Merchandise Hierarchy Levels (ASH_MHL_TBL)	Eight levels: Chain Company Division Department Class Style Color Product Key (Chinese value for)
Location Hierarchy Levels (ASH_LHL_TBL)	Five levels: Chain Company Zone Price zone (Chinese value for) Store

Table 3–6 (Cont.) Price Dataset Data

Data Interface Table	Data Characteristics
Cross Products Information (ASH_CP_TBL)	MH-LH intersections: Optimization Product key (merchandise level - Chinese value)) Store Worksheet Department (merchandise level) Chain (location level) Sales Product key (merchandise level-Chinese value) Store (location level) Cluster CHAIN CHAIN DEFAULTLEVEL CHAIN CHAIN
Merchandise Hierarchy (ASH_MH_TBL)	34,444 merchandise items at the product key level - Chinese value
Location Hierarchy (ASH_LH_TBL)	6 price zones (Chinese vlaue) 845 stores
Items (ASH_ITEMS_TBL)	7,789 items
Sales (ASH_SALES_TBL)	5 weeks of data
Markdowns Taken (ASH_MDTAKEN_TBL)	270 external markdowns
Promotions (ASH_PROMOS_TBL)	5,000 promotions (inclusions set as promo_price at the item level)
Distribution Center Inventory (ASH_DCI_TBL)	5 weeks of data
Distribution Center Allocation (ASH_DC_ALLOCATION_TBL)	8,400 allocations
Demand Parameters (ASH_PARAMETER_VALUES_TBL)	160 elasticity values
Price Ladders (ASH_PRICE_LADDERS_TBL)	178 price ladders
Seasonalities (ASH_SEASONALITY_MAPS_TBL) and (ASH_SEASONALITY_VALUES_TBL)	300 seasonality settings
Pricing Groups (No interface or staging table)	200 pricing groups
Business Rule Instances (ASH_BRM_INSTANCE_TBL)	3,023 instances of default rules: 509 OUT_DT 1,000 INVENTORY_TARGET 1,501 PLANNED_START_DT other

Modifying the Dataset

You can modify the dataset to change or add data. Here is a sample procedure for changing the unit cost for an item in the worksheet.

1. Load the staged files into the ASH tables.

   cd *<destDir>/dataset*

   bash ./pl_stage_client.sh *full_path_to_product_ directory PriceDataset*

2. Identify the target table for the item you want to modify. You can trace back from the P4P_DISPLAY_ITEMS view to the ITEM_DATA table (which supplies the unit cost data) to the ITEMS_TBL table (which populates ITEM_DATA). So, the unit cost data must be changed in ASH_ITEMS_TBL.
3. Determine (from ASH_ITEMS_TBL) the merchandise key and the location key for the ITEMID whose unit cost you are changing.
4. Create a script to apply the change directly to the ASH stage table.

   Here is a sample script called **pl_update_dataset.sql**.

   ```
   UPDATE ash_items_tbl
        SET unit_cost=16.02
   WHERE merchandise_key='10000009'
        AND location_key='5';
   COMMIT
   ```
5. Run the following script to update the changed record in ASH_ITEMS_TBL.

   bash ./pl_update_client.sh *[filename]*

   where filename is the name of the SQL script file. (This defaults to **pl_update_dataset.sql** if not specified.)
6. Stage and load the data.
7. To verify that the change was made, view the item in P4P_DISPLAY_ITEMS or in ITEM_DATA.

Sample Model Run Results

Here is a sample of expected model run results using the dataset and an "out-of-the-box" configuration:

```
Stoplight Summary
Yellow     18048
Green       3690
```

Message Breakdown

Category	Resource	Item Count	Message
Item Data	engine.status.agorai.badLastStoreCount	8136	Last week of historical activity does not have a good Store Count
Item Data	engine.status.agorai.noCandidatePrices	893	No candidate prices available for markdown: filtered Price Ladder is empty
Item Data	engine.status.didyma.noStartDate	146	This item has no Start Date
Item Data	engine.status.didyma.noTicketPrice	146	This item has no Ticket Price
Item Data	engine.status.agorai.effectiveDateMustPrecedeOutDate	9	OutDate precedes Effective Date
Inactivity	engine.status.agorai.noUsefulSales	6786	Data too dirty to determine demand: zero useful sales found

Error	engine.status.agorai.collectionItemErrors	1786	Pricing Group contains items with errors
Error	engine.status.didyma.itemBuild	74	Item contains other errors
Error	engine.status.didyma.collectionBuild	72	Pricing Group contains other errors
Markdown Blocked	engine.status.agorai.noMarkdownPricesAtEffective Date	58	No markdown prices available on Effective Date
Not Recommended	engine.status.agorai.sellsOutWithout Changes	1794	Sells to target without changes; Markdown not recommended
Not Recommended	engine.status.agorai.notRecommended	1266	Markdown not recommended for this week
Warning	engine.status.agorai.priceAboveFull	572	A relative price is higher than the full price: 1.001

4

The Model Run

This chapter contains the following:

Introduction

This chapter provides an overview of the entire model run process and details the steps of the optimization run process.

The weekly batch process is the entire process of preparing for an optimization run, including loading data and customizing Price parameters, performing the pre-optimization run steps, and performing the post-optimization run steps.

During an optimization run, Price analyzes business data and produces markdown recommendations and forecasts.

The Optimization Engine

The Optimization Engine (also known as the Calc Engine) is the software that computes optimizations and forecasts for Price. Functionally, and at a high level, the Optimization Engine consists of Delphi, which is responsible for all database operations, chunk management, the writing of results back to the database, and the calculation of forecast parameters; Agorai, which is responsible for complete forecasts and for determining optimal pricing strategy; and an RMI server, which exposes the optimization engine functionality for use by What If simulations.

The optimization engine processes are managed according to the following model. A job, which is identified by a job ID, is a logical unit of work that occurs over a defined period of time. For example, the weekly batch run process, consisting of the Price

optimization run and resultant database changes, is considered a job. Only one job can be open at a time and it remains open until it is explicitly closed using **closeCurrentJob.sh**. Keeping a job in an open state permits What If simulations to occur and also allows for the debugging of the batch process.

Once a job is opened, it is assigned a job ID and a work queue is generated. A group of one or more worker processes is started to process the work in the work queue. A worker process consists of three threads: one thread obtains the work chunk (a set of items or collections to be optimized) located at the top of the work queue (that is, with the highest priority), a second thread checks for an infinite loop, and a third thread acts as a heartbeat. If a process fails (for example, because of an infinite loop) its chunk returns to the work queue, but with a lower priority, since the heartbeat will not update the claim on that chunk. The thread responsible for the item processing obtains values via the inference rules, sends a function call to Agorai requesting an optimization calculation, and writes the results to the database. Once a worker process completes a chunk of work, it obtains the next chunk in the queue until all of the chunks have been processed.

Three types of errors can occur during a job: job initialization failures, worker process errors, and item optimization errors. Item-level errors are written to the database in the rtm tables.

Optimization Engine Configuration

The following, found in <ConfigRoot>/Engine, should be configured:

Settings for job.properties

- Database credentials
- **chunk.tryLimit** - defines the maximum number of times that the engine tries to process an item before deciding that the item cannot be processed. This value must be set to a value greater than 1. The default reflects the optimal policy according to simulations.
- **chunk.sizes** - defines a sequence of values representing the sequence of chunk sizes that should be used to group items that have had 0, 1, 2... retries. The sequence must
  - consist of a decreasing set of positive integers
  - equal in length the value of chunk.tryLimit
  - end with a value of 1
  - use semicolons to separate the series of values

  The sequencing is used in a progressive manner, starting with a large chunk and retrying with smaller chunks, to determine which items are causing the chunk to fail. (Retries are seldom needed; they happen when processes die.)

  The processing of large chunks takes up much of an optimization run, so the size of largest chunk has an impact on performance. The default sequence of values is recommended for good performance.

  When selecting the largest chunk size, consider the following. Since larger chunks require more random access memory per worker, at some point processes will either fail or use virtual memory paging. Smaller chunks are better because they incur less overhead in database access. It is suggested that the largest chunk size be a value between 1,000 and 10,000. The smaller chunk sizes should be much smaller than the largest value in order to capture work before a failure occurs.

Default value = 10,000; 100; 1

- **worker.lifetime** - defines how long in minutes the processor is allowed to run before it is decided that it is in an infinite loop and terminates. This value must be greater than 1 minute. For large collections, it is recommended that you start with a setting greater than 30 minutes.

  Default value = 30

- **chunk.active** - defines the maximum time in minutes after a worker is killed that the chunk it was working on can be reclaimed. This value must be greater than 1 minute. This setting rarely needs customizing.

  Default value = 3

Settings for delphi.properties

The delphi.properties file should only list values that differ from the default values. If a default value exists, it is listed here. The first two properties, for the Agorai library location and the RMI server port, are required. All others are optional.

Table 4–1 delphi.properties Settings

Property	Description
engine.agorai.lib=installer-supplied-path/libAgoraiJNI.so	The location of the Agorai library.
delphi.rmi.port=installer-supplied-port-number	The port used by the RMI server in interactive mode.
batch.write.size=100	The number of items or collections contained in a batch written to the database.
optimize.status.tbl=item_status_tbl	The table the optimization status for each item or collection is written to.
engine.record.directory=pathname	The path of the message capture directory. In order for messages to be logged, a complete path and an existing directory are necessary.
engine.record.internals=false	Set this property to true in order to log internal engine messages. (This assumes engine.record.directory has been created and enabled.)
delphi.log4j.properties=delphi.log4j.properties	The file that contains properties for controlling Delphi logging behavior. The path is relative to the path for delphi.properties.
strategy.activitydata=list	Used for performance tuning of IR_ACTIVITY_DATA. Values are single, list, and temptable. *See* Inference Rule Access for more information on all strategy properties.
strategy.businesspolicy=list	Used for performance tuning for IR_BUSINESS_POLICY.
strategy.distribution=list	Used for performance tuning for IR_PRIOR_DISTRIBUTION.
strategy.forcedmarkdowns=list	Used for performance tuning for IR_FORCED_MARKDOWNS.
strategy.itemdates=list	Used for performance tuning for IR_ITEM_DATES.
strategy.itemparameters=list	Used for performance tuning for IR_ITEM_PARAMETERS.
strategy.itemprices=list	Used for performance tuning for IR_ITEM_PRICES.
strategy.markdowncalendar=list	Used for performance tuning for IR_MARKDOWN_CALENDAR.

Table 4–1 (Cont.) delphi.properties Settings

Property	Description
strategy.modelvalues=list	Used for performance tuning for IR_MODEL_VALUES.
strategy.pastticketprices=list	Used for performance tuning for IR_PAST_TICKET_PRICES.
strategy.pendingmarkdowns=list	Used for performance tuning for IR_PENDING_ MARKDOWNS.
strategy.plannedpromos=list	Used for performance tuning for IR_PLANNED_PROMOS
strategy.priceladder=list	Used for performance tuning for IR_PRICE_LADDER.

Optimization Run Prerequisites

Once Price is installed, the following must be configured prior to an optimization run.

- Merge any existing customized load statements with the updated **load_ statements.sql**. The load statements are used for eligibility filtering and for populating the ITEM_DATA table. **Load_statements.sql** is installed under **db.config**.
- Merge any existing customized inference rules (located in **ir.sql**, installed under **db.config**) with the updated inference rules.
- Apply **load_statements.sql** and **ir.sql** to the database schema, using either **configdb.sh** or **plconfiguredb.sh**, as follows:
  - copy **ir.sql** and **load_statement.sql** to **$CONFIGROOT/db.config**
  - **cd $PL_BASE/modules/tools/bin**
    where **PL_BASE** is the location where Price is installed.
  - **bash plconfiguredb.sh $CONFIGROOT**
- Business Rule Definitions - Price comes with default values for BRs, set at the highest level (except Outdates and planned Start Dates). Merge any customized business rule definitions with the updated version.
- Business Rule Values - load using bulkloader.

Note: No special maintenance of tables or indexes is required. All necessary statistics gathering and index rebuilding is handled by the application.

Note: The performance of the optimization run is affected by the number of candidate prices from the price ladder (those lower than the current price) and candidate markdown calendar (which depends on how far out the outdate is). Performance should be considered when choosing prices for the price ladder; if the price ladder is long, performance may be adversely impacted.

Optimization Run Process

The weekly Optimization Run consists of the following high-level steps. These steps are all executed by **weeklyBatch.sh**. (The details about each step are provided in subsequent sections.)

Note: Since the model run and KPI share work_queue_tbl, you should not run KPIs and the model run at the same time.

1. **plfrontendload.sh**, which executes FELOAD
2. **plpremodelrun.sh**, which executes PRERUN

3. **runCalcEngine.sh**, which executes a series of helper scripts responsible for the batch process
4. **runMultiKPI(Item | Collection).sh**, which calculates the key performance indicator metrics
5. **plpostmodelrun.sh**, which executes POSTRUN
6. **refreshSummaryCache.sh**, which refreshes the P4P_WORKSHEET_SUMMARIES cache table
7. **refreshForecastCache.sh**, which refreshes the forecast cache

Optimization Run Scripts

This section contains details about the scripts used during an optimization run, listed in alphabetical order.

bashjava.sh

Usage: **bashjava.sh**

Description:

A shell script wrapper around Java that is used by the installer.

checkKPISuccess.sh

Usage: **checkKPISuccess.sh** *<full_path_to_product_directory>*

Description:

Checks to see if the KPI calculations have been performed for all items and all items in collections. It prints the number of items not calculated. It returns a value of 0 if the KPIs have been calculated for all items. It returns a value of 1 if some calculations are missing.

checkRunSuccess.sh

Usage: **checkRunSuccess.sh** *[-dh] <full_path_to_configroot>*

where

- d - debug mode (turns on Java asserts)
- h - print this message and exit

Description:

Checks to see if the optimization run completed successfully. It returns a value of 0 if the command completes successfully; however, this exit code provides no information on the status of the optimization run itself.

The output of this command looks similar to the following:

```
Success: Run is complete.
Missing: 0 out of 15578, Errors: 2
```

If the number of optimization is equal to the number of eligible items, then the command outputs “Success: Run is complete.” If the number of optimizations does not equal the number of eligible items, then the command outputs “Failure: Run is not complete.” The number indicated for Missing is the difference between the number of optimizations and the number of eligible items. The number indicated for Errors is the number of items that have a status of Red.

closeCurrentJob.sh

Usage: **closeCurrentJob.sh** *[-dh] <full_path_to_configroot>*

where

- d - debug mode
- h - print this message and exit

Description:

Called by **runCalcEngine.sh**. Closes the current job, if one exists. As a result, the worker processes on the batch process exit.

This command closes the job record in the job control database. All worker processes frequently check that their current job is still open, and they exit if it is not. When they exit is such a situation, the auto-restart feature does not restart the process. This command can be executed from any machine and all workers on all machines will shut down.

The commands **kill** and **killall** cannot be used to kill the worker processes, as they will restart.

enginectl.sh

Usage: **enginectl.sh** *<full_path_to_configroot> (start | stop | kill | restart | status | help)*

where

- start - start the interactive Optimization Engine
- stop - stop the interactive Optimization Engine
- kill - stop the interactive Optimization Engine
- restart - try to stop and then restart the Optimization Engine
- status - display the status
- help - print the usage message

Description:

Starts and stops the RMI server, which is used for What If simulations. For more information, see the Price Configuration Guide.

generateErrorWorkQueue.sh

Usage: **generateErrorWorkQueue.sh** *[-dh] <full_path_to_configroot>*

where

- d - debug mode (turns on Java asserts)
- h - print this message and exit

Description:

Compiles a list of all items and collections that have failed during the current batch job. Once the problems in this list have been corrected, the items and collections that failed can be retried.

getCurrentJobID.sh

Usage: **getCurrentJobID.sh** *[-dh] <full_path_to_configroot>*

where

- d - debug mode (turns on Java asserts)
- h - print this message and exit

Description:

Called by **runCalcEngine.sh**. Returns the ID of the current batch job to stdout.

getCurrentJobStatus.sh

Usage: **getCurrentJobStatus.sh** *[-dh] <full_path_to_configroot>*

where

- d - debug mode (turns on Java asserts)
- h - print this message and exit

Description:

Called by **runCalcEngine.sh**. Returns an integer value between 0 and 100 that represents the percentage of the current batch job that has been completed.

getEngineVersion.sh

Usage: **getEngineVersion.sh**

Description:

Prints the build version of the Optimization Engine to ***stdout***.

initializeJob.sh

Usage: **initializeJob.sh** *[-dh] <full_path_to_configroot>*

where

- d - debug mode (turns on Java asserts)
- h - print this message and exit

Description:

Called by **runCalcEngine.sh**. It initializes a new batch job and prints the job ID of the batch job to stdout. It returns a value of 0 if the initialization is successful. If the initialization fails, it prints a reason to ***stdout***.

isDone.sh

Usage: **isDone.sh** *[-dhv] <full_path_to_configroot>*

where

- d - debug mode (turns on Java asserts)
- h - print this message and exit
- v - verbose mode

Description:

If the batch job is complete, it returns a value of 0. If the batch job is not complete, it returns a value of 1.

jobHistory.sh

Usage: **jobHistory.sh** *[-dh] <mhNode> <full_path_to_configroot>*

where

- d - debug mode (turns on Java asserts)
- h - print this message and exit

Description:

It provides a detailed report about the current batch job.

jobReport.sh

Usage: **jobReport.sh** <*full_path_to_configroot*>

Description:

Called by **runCalcEngine.sh**. It provides a summary report about the current batch job.

multiChunker.sh

Usage: **multiChunker.sh** *[-dh] [-n num_threads] <full_path_to_configroot>*

where

- d - debug mode (turns on Java asserts)
- h - print this message and exit
- n - number of threads to be run

Description:

It starts several batch worker processes. If any of its child processes returns a value of 0 (that is, completes successfully), the script itself returns a value of 0. In other words, if one child process completes successfully, it indicates that the entire batch process has completed successfully.

plfrontendload.sh

Usage: **plfrontendload.sh** <*configroot*> *[OracleItemDataPartitioning Flag]*

where

- configroot is the output directory for the configuration (suite's configroot)
- OracleItemDataPartitioningFlag specifies ITEM_DATA table partitioning rules in Oracle as follows:
  - -1 - disable partitioning
  - any other value or blank - do range partitioning as one partition per worksheet

Description:

Executes FELOAD.

plpostmodelrun.sh

Usage: **plpostmodelrun.sh** <*configroot*> *[NumItemDataPartitions]*

where

- configroot is the output directory for the configuration
- NumItemDataPartitions specifies the ITEM_DATA table partitioning rules as follows:

- positive integer - use this number of partitions for partitioning
- 0 - use the default number of partitions for partitioning
- -1 - disable partitioning
- any other value or blank - use the default number of partitions for partitioning

Description:

If NumItemDataPartitions is specified, this command simply stores the specified value in the P4P_PARAMS table, which is used to drive the partitioning logic when it is subsequently executed. When a fresh schema is created, this parameter is initialized (that is, seeded in the P4P_PARAMS table) to -1, which means that partitioning is disabled. However, when this script is run, this parameter will be updated according to the NumItemDataPartitions options. Note that to keep partitioning permanently disabled, you always need to specify -1 on every call to this script; otherwise, it will be reset to the default value. Schema upgrades preserve the current value of NumItemDataPartitions.

plpremodelrun.sh

Usage: **plpremodelrun.sh** <*configroot*>

where

- configroot is the output directory for the configuration (suite's configroot)

Description:

Executes PRERUN.

refreshForecastCache.sh

Usage: **refreshForecastCache.sh** -s <*server*> -t <*number of threads*> *[-c]*

where

- s - the url of the remote application server
- t - the number of threads being used
- c - processing continues on failure

Description:

The **refreshForecastCache.sh** script refreshes the forecast cache. It operates synchronously. It replaces pinging the WorksheetForecastServlet.

refreshSummaryCache.sh

Usage: **refreshSummaryCache** *-s*

Description:

The **refreshSummaryCache.sh** script causes summary metrics to be recalculated based on current values in the ITEM_DATA table. It should be invoked after every model run or when the configuration changes.

resetForKPIItem.sh

Usage: **resetForKPIItem.sh** <*full_path_to_product_directory*>

Description: During the optimization run, Job Control keeps track of which chunks have been completed. The script resets the Job Control system after the optimization run and before the KPIs are calculated. It clears the Job Control database table and

populates a KPI status table with the item IDs that will have KPIs calculated. This script should be run before **runMultiKPI.sh** for items.

resetForKPICollection.sh

Usage: **resetForKPICollection.sh** *<full_path_to_product_directory>*

Description: During the optimization run, Job Control keeps track of which chunks have been completed. The script resets the Job Control system after the optimization run and before the KPIs are calculated. It clears the Job Control database table and populates a KPI status table with the collection IDs that will have KPIs calculated. This script should be run before **runMultiKPI.sh** for collections.

runCalcEngine.sh

Usage: **runCalcEngine.sh** *[-n numWorkers] <full_path_to_product_directory>*

Description:

The **runCalcEngine.sh** script calls the following scripts, which are required to complete a batch run. Running the Optimization Engine across more than one application host requires running the scripts comprising **runCalcEngine.sh** independently. Only **multiChunker.sh** should run on more than one host at a time.

- **getCurrentJobID.sh**
- **closeCurrentJob.sh** (if a previous job exists)
- **initializeJob.sh**
- **multiChunker.sh**
- **getCurrentJobStatus.sh**
- **runStatsOnBatchOutput.sh**
- **jobReport.sh**

runChunker.sh

Usage: **runChunker.sh** *[-dh] <full_path_to_configroot>*

where

- d - debug mode (turns on Java asserts)
- h - print this message and exit

Description:

Starts a single worker process in the foreground.

runInteractiveCE.sh

Usage: **runInteractiveCE.sh** *[-p] <full_path_to_configroot>*

where

- p - engine starts in failover mode

Description:

Called by **enginectl.sh**. It starts the RMI server and an associated watchdog process that restarts the RMI server if it crashes.

runMultiKPI.sh

Usage: **runMultiKPI.sh** *[-n numWorkers] item | collection <full_path_to_product_directory>*

where

- n - number of parallel processes
- item | collection - one of these arguments must be selected

Description:

The **runMultiKPI.sh** script calculates performance metrics that are based on Optimization Engine forecasts (key performance indicators). The script confirms that a job is initialized and then it runs the KPI calculations.

The KPI calculations are divided into two separate batch processes, KPI for items and KPI for collections. The script takes either the **item** argument or the **collection** argument. The KPI for items calculation *must* be completed before the KPI for collections calculation.

Before each KPI for items calculation is run, the **resetForKPIItem.sh** command must be run to reset the Job Control queues. Before each KPI for collection calculation is run, the **resetForKPICollection.sh** command must be run to reset the Job Control queues.

These calculations can be done in a single process or in multiple parallel processes.

The weekly batch script calls these scripts in the following order:

1. resetForKPIItem
2. runMultiKPI for items
3. resetForKPICollection
4. runMultiKPI for collections

See also **resetForKPIItem.sh** and **resetForKPICollection.sh**.

runReport.sh

Usage: **runReport.sh** *[-dh] <full_path_to_configroot>*

where

- d - debug mode (turns on Java asserts)
- h - print this message and exit

Description:

It prints a report of the current batch run to ***stdout***.

runStatsOnBatchOutput.sh

Usage: **runStatsOnBatchOutput.sh**

Description:

The **runStatsOnBatchOutput.sh** script is called by **runCalcEngine.sh** and is part of **weeklyBatch.sh**. The script runs an "analyze and estimate" utility call RunStats on markdown_activities, forecast_summaries, forecast_activities, rtm_history, rtm_status, and rtm_status_arguments.

weeklyBatch.sh

Usage: **weeklyBatch.sh** *[-n] <full_path_to_product_directory> <p4pgui-host> <p4pgui-port>*

where

- n - number of chunks

Example: **weeklyBatch.sh** *-n 3 /profit ps-app-l01 9040*

Description: This script encapsulates the weekly batch process. It is best used with small datasets as a tool during the Price configuration process; it is not recommended for use in a production environment.

If you do not specify the number of chunks, the optimization engine and the KPI calculations both use one thread.

Load_statements.sql

The following are contained in **load_statements.sql** and are part of the optimization run.

FELOAD

Prior to FELOAD, certain tables are cleaned up in order to improve performance. During the FELOAD portion of the optimization run, the ITEM_DATA table is updated with new data that has been received from the client.

FELOAD consists of the following steps:

1. Procedures for archiving ITEM_DATA and Forecasts are loaded.
2. The SCENARIOS_TBL table is loaded.
3. The INTERNAL_PROMO table is loaded.
4. The INTERNAL_HIST_MKDNS_TBL table is loaded.
5. The INTERNAL_OUT_DTS_TBL table is loaded.
6. All invalid views are recompiled.
7. The INTERNAL_ITEM_DATA_TBL table is truncated.
8. The INTERNAL_ITEM_DATA_TBL table is populated.
9. All dropped indexes are restored on the internal ITEM_DATA table.
10. Statistics are collected on the internal ITEM_DATA table.
11. The FEDATES tag, which includes updating P4P_PARAMS, is created.
12. The COLLECTIONS tag, which truncates the P4P_COLLECTION table, is created. This step is not called by any other step in **load_statements.sql**.
13. The P4P_COLLECTION table is truncated.

Data Archiving Data in Price is archived in such a manner that data not required for the weekly optimization run or by the Price application is cleaned up regularly. Required data is preserved and performance is enhanced.

Two procedures, which are part of the FELOAD portion of **load_statements.sql**, are responsible for archiving:

- **com.profitlogic.db.birch.ArchiveForecasts**

- **com.profitlogic.db.birch.ArchiveItemData**

Archiving occurs once a week. The current data in each table is compared to the data in the archive and only the new records are selected to be appended to the archive.

The following table lists the archiving source tables and target tables.

Table 4–2 Archived Tables

Archive Source Table	Archive Target Table	Populated By	Amount of History Preserved
FORECAST_ACTIVITIES	FORECAST_ACTIVITIES_ARCH	Model Run	unlimited
FORECAST_RUNS	FORECAST_RUNS_ARCH	Model Run	unlimited
FORECAST_SUMMARIES	FORECAST_SUMMARIES_ARCH	Model Run	unlimited
MARKDOWN_ACTIVITIES	MARKDOWN_ACTIVITIES_ARCH	Model Run	unlimited
RTM_HISTORY	RTM_HISTORY_ARCH	Model Run	unlimited
RTM_STATUS	RTM_STATUS_ARCH	Model Run	unlimited
RUN_HISTORY	RUN_HISTORY_ARCH	Model Run	unlimited
ITEM_DATA	ITEM_DATA_ARCH	FELOAD, KPIs, user actions	two years

Forecast Archiving The forecast archiving process maintains a status table that is truncated at the beginning of each weekly archiving. As each step in the process is successfully completed, its completion status is logged into the status table (ARCHIVING_STATUS_TBL). The cleanup of the source database tables, which is the last step of the procedure, begins only after all required archiving steps have been completed.

ITEM_DATA Forecasting The ITEM_DATA forecasting step preserves all the columns from the ITEM_DATA table. The ITEM_DATA_ARCH table contains one additional column, CURRENT_WEEK, which is populated from the PARAM_NAME column (= CurrentWeek) in the P4P_PARAMS table. The ITEM_DATA archiving step included the truncating of the ITEM_DATA table.

PRERUN

During the PRERUN portion of the optimization run, the BRM business rules are cached into ITEM_BRM_RULES at the item level. Each column in the table represents a separate business rule (in correspondence to BRM_RULE_DEFINITION TBL). Each row in the table represents a unique ITEM_ID, with its business rules exploded to the item level. The ITEMS view along with the SEASONALITY_ID are cached in ITEMS_MODELRUN_TBL, which is used as a source for most inference rules.

Note: Chunking has been disabled by default in the BRM caching step. Chunking should only be enabled if processing is too slow. To enable chunking, configure the BRMChunkSize parameter in P4P_PARAM as follows:

Table 4–3 BRMChunkSize Parameter Values

BRMChunkSize Value	Description
0	No chunking (the default)

Table 4–3 (Cont.) BRMChunkSize Parameter Values

BRMChunkSize Value	Description
> 0	Chunks created of size specified
< 0	Chunking not allowed

The value can be set manually or by using the script **..\modules\tools\bin\plsetbrmchunksize.sh**. The value is set to the default value of 0 at the time of the initial load.

PRERUN consists of the following steps:

1. The BRM business rules are cached into the ITEM_BRM_RULES table.
2. All indexes are dropped on the ITEM_MODELRUN_TBL table.
3. The ITEM_MODELRUN_TBL table is truncated.
4. All What If output tables are truncated.
5. The RECALC_OVERRIDE_TBL table is populated with the required row of seed data.
6. FULL_PRICE and PRICE_LADDERS are cached for items.
7. The ITEM_MODELRUN_TBL table is populated.
8. All dropped indexes are restored on the ITEM_MODELRUN_TBL table.
9. The IR_WAREHOUSE_CACHE_TBL, which is a cache for IR_WAREHOUSE, is loaded.
10. BRM_ATTRIBUTE_VALUE_TBL is populated.
11. HIST_MARKDOWNS_MODELRUN_TBL (HIST_MARKDOWNS cache) is populated.
12. The PERIODS_MD_CAL_TBL table, which is a subset of PERIOD_TBL, used by the IR_MARKDOWN_CALENDAR view, is populated.
13. The P4P_LADDER_ROLES_TBL is populated.
14. All table statistics are updated.
15. All invalid views and schemas are recompiled.

POSTRUN

During the POSTRUN portion of the model run, the P4P_FORECAST_DATA table is updated with the latest model run results, the ITEM_DATA table is updated with certain metrics, and RECOMMENDED_COLLECTION_FLAG and WORKSHEET_ID in P4P_COLLECTIONS are updated with data from the ITEM_DATA table.

POSTRUN consists of the following steps:

1. All indexes are dropped on the P4P_FORECAST_DATA table.
2. The P4P_FORECAST_DATA and TMP_P4P_FORECAST_DATA tables are truncated.
3. The TMP_P4P_FORECAST_DATA table is loaded.
4. Statistics are collected on the P4P_FORECAST_DATA table.
5. The P4P_FORECAST_DATA table is loaded.

6. All dropped indexes are restored on the P4P_FORECAST_DATA table by the RestoreTable procedure.
7. Statistics are collected on the P4P_FORECAST_DATA table.
8. The TMP_P4P_FORECAST_DATA table is truncated in preparation for the load.
9. All indexes are dropped on TMP_POSTRUN_ITEM_DATA.
10. The TMP_POSTRUN_ITEM_DATA table is truncated.
11. The temp table used in the ITEM_DATA update to populate proj_oh_units_eff_dt is loaded.
12. All dropped indexes on TMP_POSTRUN_ITEM_DATA are restored using the RestoreTable procedure.
13. Statistics are collected on the TMP_POSTRUN_ITEM_DATA table.
14. The COLLECTION_NAME, PROJ_OH_UNITS_EFF_DT, and RECOMMENDED_COLLECTION_FLAG columns in ITEM_DATA are updated, based on the most recent model run results.
15. Statistics are collected on the ITEM_DATA table.
16. The P4P_COLLECTION.RECOMMENDED_COLLECTION_FLAG column is updated with the latest information from ITEM_DATA.
17. Statistics are collected on the P4P_COLLECTION table.
18. All invalid schema objects are recompiled.
19. All invalid views are recompiled.
20. The ITEM_DATA_CUR_METRICS_TBL is truncated in preparation for its load.
21. The ITEM_DATA_CUR_METRICS_TBL is loaded with all eligible items from ITEM_DATA. This allows P4P_DISPLAY_ITEMS to obtain current and recommended metrics using regular joining.

Summary Metrics

Note: Materialized views are not supported and should be removed.

The P4P_WORKSHEET_SUMMARIES table stores the aggregate data for all summary metrics for all worksheets. The P4P_WORKSHEET_SUMMARIES table is indexed after the data is populated, and the name of the index is SummaryCache_IDX.

This table is initially created with one column, the worksheet_ID column, when the application is deployed. The Worksheet Summaries page in Price obtains summary metrics from P4P_WORKSHEET_SUMMARIES. Parameters associated with the worksheet summary cache are stored in P4P_SUMMARYCACHE_PARAMS. The parameters table should always exist. The summary metrics table can be rebuilt or refreshed.

When P4P_WORKSHEET_SUMMARIES table is rebuilt, the existing table is dropped, then recreated and populated with data. When the table is refreshed, all rows in the table are deleted and re-populated.

A refresh of the cache table is triggered when

- the application server is running and the **refreshSummaryCache.sh** command is invoked

The cached table is rebuilt when:

- the application server is running and the configuration has changed and the **refreshSummaryCache.sh** command is invoked.
- The configuration has changed and the application server is restarted.

If the application server is restarted but the configuration has not changed, then the cache table is not rebuilt or refreshed. So, P4P_WORKSHEET_SUMMARIES should be refreshed/rebuilt after every model run and whenever changes are made to xml files in the p4pgui/grids directory, or if a change is made from the UI that affects the summary metrics, i.e., taking a markdown via a worksheet.

In addition, grids are refreshed through **p4padmin.jsp** when the application server is running. Clicking on the Grid Configuration link reloads the grids and rebuilds the summary metrics cache.

A link, Worksheet Summary Cache Information, provides the following diagnostic information:

- the names of the cached columns
- the worksheet IDs that are cached
- the SQL statement used to calculate the summary metrics

Monitoring an Optimization Run

The commands you can use to monitor the progress of the optimization run include:

- **getCurrentJob.sh**
- **getCurrentJobStatus.sh**
- **isDone.sh**
- **jobHistory.sh**
- **jobReport.sh**
- **runReport.sh**

The optimization run can be monitored by reviewing the exit codes for the worker processes from **runCalcEngine.sh** and **multiChunker.sh**.

Resource monitoring of the application host that the worker processes are running on and the database host that the worker processes communicate with is also recommended. The saturation or overuse of hardware can indicate a configuration problem, such as the wrong number of worker processes, the wrong number of worker processes per machine, the wrong chunk size, or inappropriate heartbeat times.

The **isDone.sh** utility returns an exit code of 0 if the current batch run is complete; otherwise, it returns an exit code of 1.

The **getCurrentJobStatus.sh** utility prints a number between 0 and 100, which represents the approximate percentage of processing completed. This value is computed as a weighted percentage of completed chunks from the work queue, so the value is less accurate if business rules are more heterogeneous across merchandise or if the chunks are large.

The **jobReport.sh** utility prints a detailed breakdown of the number of items completed, the number of collections completed, and the number of optimizations of each type that have failed.

The **runReport.sh** utility prints the Stoplight Summary. This is available at any time during the batch process, but it does not indicate if the job is complete.

These monitoring scripts provide a database-level view of how the run is proceeding. However, monitoring the exit status of the worker processes for unexpected failures is also recommended. These unexpected failures may indicate a configuration or data problem such as overly aggressive suicide times or problems with inference rule customization. It is also recommended that you redirect ***stderr*** to a log file in order to view any warning messages.

Running Reports and Diagnosing Problems

The **runReport.sh** utility prints the Stoplight Summary. It is available at any time during the batch job, regardless of the value returned by **isDone.sh**.

The Stoplight Summary section of the report provides a count of all errors, categorized by the level of severity of the error. The red category indicates system or configuration errors that must be fixed. The yellow category includes errors that result from missing data or a possible mis-configuration of the application. The green category indicates no errors.

The Message Breakdown section of the report lists the errors in the order of severity and provides a count of the number of items affected.

If items or collections are missing, the run can stop prematurely or have difficulty writing its results to the database. If this is the case, you can check the application server logs and database logs to determine the cause.

If the problem is not caused by missing data, you should first look at system and model configuration errors (Red category). You should diagnose and fix these problems and then restart using **generateErrorWorkQueue.sh** or run a completely new job.

The frequency of errors in the various categories may supply information that can be useful in diagnosing problems.

If worker processes exit prematurely with an exit code greater than zero, you should examine the item-level status and error messages with **runReport.sh**. You should also review the **worker.lifetime** setting to ensure that it provides enough time for even the longest-running pricing group optimizations. Overly aggressive settings for **worker.lifetime** can cause workers to exit abruptly.

The data structure that Delphi gathers from the inference rules and feeds to Agorai, as well as Agorai's responses, can be saved as an XML file. These messages, which contain the precise information sent to and returned from Agorai, are useful in determining whether a problem is with the data itself or has resulted from a mis-configuration (such as in the business rules).

To save these messages to an XML file, in the **delphi.properties** file set **engine.record.internals=true** and specify a complete path and directory name in **engine.record.directory**. (See "Settings for delphi.properties" on page 3 for more information.) Changes to these settings apply when the worker processes or the RMI server are restarted.

The ***stderr*** stream, if redirected to a log file, can also yield information about the cause of an exit and whether or not it involved worker.lifetime.

Restarting a Run

If some optimizations fail during the optimization run, you should correct any problems that affected the job and then re-execute the parts that had previously failed. Generate an error work queue, which contains the chunks for which one or more items

or collections reports a failure. The chunks themselves contain only those items or collections that have failed. Run **generateErrorWorkQueue.sh** in order to mark these chunks as available for processing again. Then restart the optimization run as normal.

Performance Considerations

The items in a batch run are grouped into chunks, which are processed together. This design improves performance; however, if an item fails because of a programming or configuration error, then the failure affects the processing of the entire chunk. This situation can be addressed as follows.

Configuration

Three parameters in **job.properties** affect this issue: **chunk.sizes, chunk.tryLimit,** and **worker.lifetime**. These parameters are discussed earlier in this chapter. In most cases, the default values for these properties are adequate.

Automatic Restart

Worker processes that terminate before completion are automatically restarted via the **runChunker.sh** script. It restarts processes that have exited with a status *other than* 0 (success) or 1 (unrecoverable failure). This convention should be followed if other custom scripts are used.

Stopping Jobs

Use **closeCurrentJob.sh**, described earlier in this chapter, to stop all worker processes. A new batch job can be started before the old one has completely stopped.

Note: The **kill** and **killall** commands cannot be used to stop worker processes.

Worker Restart Messages

A worker start-up message that displays at the start of a run is a normal occurrence. A worker start-up message during a run indicates that a failure has occurred.

Worker Time-outs

Worker processes detect items whose processing is taking longer than expected. In such a case, a message will be displayed indicating that the worker process has timed out and is being restarted to process the remaining items.

Problem Analysis

The failure of an item does not necessarily indicate that the entire run is bad; however, the run may take longer to complete.

To determine how much of the run has completed, use getCurrentJobStatus.sh, which is described earlier in this chapter. The command prints the percentage of the job that is complete and the amount of time that has elapsed. This information can be used to project the completion time for the run.

Failure Diagnosis

To determine what has gone wrong with a run that has completed, do the following:

1. Locate an item that has a status of *missing* (which indicates that the item has not been processed) using the following query:

```
select *
from item_status_tbl t, item_status_labels_tbl |
where t.status_id = l.status_id and l.status_desc = 'open'
```

2. To obtain a description of the item from ITEM_DATA, use the following query:

```
select *
from item_data
where item_id in (list from above query)
```

3. Capture the XML file for the item and analyze.

Job Controller

When a batch job is terminated, any worker processes that are working on that job should stop updating it. This is necessary because, if a new job is started and new batch workers are created, then the updates should apply to the new job rather than the job that has just been terminated. It is possible that a worker from an old job can live long enough to save changes into a new job, which is a bad outcome.

Performance

Production performance can vary, depending on a specific client customization.

To identify performance problems:

1. Measure performance. Determine whether the database hardware is saturated. It should be close to 100% CPU utilization. If it is not, worker processes can be added to increase throughput.
2. If CPU utilization is at 100% and the batch process cannot be completed within the production window, then verify that the database server (hardware and software) is configured correctly.
3. If the database is configured correctly, determine which Inference Rule is taking the most time by logging into the database server as administrator and examining the execution profile. An IR that is dominating the execution profile may need to be reconfigured.
4. The strategy used to query the database can be configured for an individual IR. For more information, see the Inference Rule chapter in the Price Configuration Guide.
5. The chunk size setting in job.properties can be modified. This is discussed earlier in this chapter.

Sendback Files

A sendback file is a file that is created using the PriceAdmin command, **generateSendback**. The sendback file contains information about changes made via the Price user interface. The file is the mechanism for transmitting the updated markdown information to the customer. The content of the sendback file is determined by the SQL query.

Usually, a sendback file is generated once at week at the Sendback Date, which occurs at the Cutoff Date and Time. However, the schedule for generating and transmitting a sendback file is determined by the needs of the business. Individual retailers may require sendback files at regular intervals during the week or may even require daily sendback files. The script to automate the sendback process should be designed to manage the schedule.

Sendback File Example

The following example shows the generation of a sendback file with a typical weekly schedule, called the markdown cycle:

Note: Only one sendback file is included in this example; however, retailers may require the generation of more than one sendback file as part of their standard schedule.

To generate a sendback file, you must:

- Write one or more SQL queries against the database. A query should specify the name (type) of the sendback file, line ending, file delimiter, and information to be included in the sendback file, such as item, location, value of the new price after markdown, and date of price change. Edit the **p4pgui-config.xml** file in the configuration root directory to include the necessary queries

  The following sample query generates information about outdates:

  ```
  <sendback-query name="pl-outdates"
  line-endings="unix" result-delimiter="|"><![CDATA
  [select item_id, modified_out_of_stock_date from
  p4p_items where modified_out_of_stock_date is not
  null]]></sendback-query>
  ```

- Edit the property, **p4pgui.sendback.dir = pathname** in the **configuration_root/p4pgui/config.properties** file to specify the destination of the generated sendback file.
- Create a script to automate the scheduled weekly generation of the sendback file that includes the necessary PriceAdmin commands.

Accelerated Markdowns

Since the implementation of a markdown decision as a price change visible to the retail customer can take weeks, it is possible to override the markdown schedule by taking accelerated markdowns. Under such a plan, users assign a new Sendback Date to selected items and a sendback file that captured this information is generated on every day that end users have access to the Price user interface, regardless of whether or not end users have made changes. This accelerated markdown process allows a retailer to take markdowns quickly. For more information on how accelerated markdowns are implemented in the Price user interface, see the Price User Guide.

Only one accelerated markdown is allowed for any item within a single markdown cycle. So an end-user can add or modify the sendback date for an item until the appropriate worksheet is approved and executed. The item is exported to the sendback file and the Sent Date, Sent Markdown Price, and Sent Price Ladder are updated in the database table. After this, no other accelerated markdowns are permitted for an item until the end of the current markdown cycle, when the week's sales information is loaded into the Price database, and the fields for Sendback Date, Sent Date, Sent Markdown Price, and Sent Price Ladder are cleared. These three fields are used to record the date and time that the sendback file was generated, the approved markdown price of the item at the time the sendback file was generated, and the price ladder ID of the item at the time the sendback file was generated.

To implement accelerated markdowns:

1. Write an SQL query to create an accelerated sendback file. A query should update Sent Date, Sent Markdown Price, and Sent Price Ladder in the database. Edit the **p4pgui-config.xml** file in the configuration root directory to include any queries.
2. Create a script to automate the daily accelerated markdown sendback process. Such a process should guarantee that the sendback files are generated. In addition, the process should include a fail safe so that if, for example, the sendback file generation process fails on Wednesday, then the Thursday accelerated markdown

sendback generation process picks up any unsent items that have a Sendback Date of Thursday or earlier.

Note: The regular weekly sendback script should include functionality to pick up any unsent items.

Accelerated Markdown Sendback Examples

Here is an example of an accelerated markdown sendback query that collects all accelerated markdowns from the day (or any that have not been collected since the last cutoff):

```
<sendback name="client-markdowns-daily">

<select-query line-endings="unix"
result-delimiter="|"> <![CDATA [select i.hierarchy3,
i.hierarchy6, i.proposed_price,
to_char (w.timestamp_approve, 'mm-dd-yyyy hh24:mi:ss')
from p4p_items i, P4P_SUBMITTAL_WORKSHEETS w WHERE
i.SUBMITTAL_WORKSHEET_ID = w.SUBMITTAL_WORKSHEET_ID AND
i.sent_date=SYSDATE]] > </select-query>

<pre-sendback-update > <![CDATA[UPDATE p4p_items SET
sent_ladder_id=ladder_id, sent_markdown_price =
proposed_price, sent_date = SYSDATE WHERE markdown_flag
> 0 AND sent_date IS NULL AND TRUNC (sendback_date) <=
SYSDATE AND submittal_worksheet_id IN (SELECT
submittal_worksheet_id FROM P4P_SUBMITTAL_WORKSHEETS
WHERE worksheet_status_id = 3)]]> </pre-sendback-update>

</sendback>
```

To create the sendback file, use the following command:

PriceAdmin.jar generateSendback *-t client-markdowns-daily -o filename.txt*

The file is overwritten each time the query runs, so rename the file if you want to keep a record of previous daily markdown files.

Here is an example of an accelerated markdown sendback query that collects all non-accelerated markdowns and all accelerated markdowns that were not collected since the last cutoff date:

```
<sendback name="client-markdowns-weekly">

<select-query line-endings="unix" result-delimiter="|"><![CDATA[SELECT
item-id, proposed_price FROM p4p_items i, P4P_SUBMITTAL_WORKSHEETS
sw WHERE i.submittal_worksheet_id=sw.submittal_worksheet_id AND
i.markdown_flag='y' AND sw.worksheet_status_id=3 AND i.sent_date IS
NULL>]]>

</select-query>

</sendback>
```

To create the sendback file, use the following command:

PriceAdmin.jar generateSendback *-t client-markdowns-weekly -o filename.txt*

The file is overwritten each time the query runs, so rename the file if you want to keep a record of previous weekly markdown files.

Automating Price Processes

The following Price processes occur regularly. Most of these processes consist of a sequence of steps. These processes may be suited to scripting/automation and scheduling using an entrerprise scheduler. Some processes, such as the standard load and the optimization run, may benefit from a granular, step-wise process in which each step returns an exit code upon completion. Such a design can help with troubleshooting and recovery.

- Standard Load. Price provides two scripts that stage, transform, and load data into the target database tables in the Price database. See Chapter 3, "Standard Load" for details.
- Optimization Run. This process takes the raw input data from the client, runs the data through the forecasting and optimization engine, and writes the results to database tables that are read by the Price front end application. This process is discussed in detail in this chapter.
- RDM Updates. If the Retail Data Mart is being used, it must be updated with current Price data regularly. See Chapter 5, "RDM Loads" for details.
- Alerts Generation. If the Merchant Desktop and alerts are being used, alerts must be generated. See the Price Configuration Guide for details.
- Sendback Generation. Sendback files are scheduled according to a client's business needs. Sendback generation is discussed in detail in this chapter.

In addition to these standard processes, most Price implementations include customized processes that are tailored to specific customer requirements. The most common of these involve custom sendback feeds in which a specific file or set of files must be generated in order to feed data to another system.

5

RDM Loads

This chapter contains the following:

Introduction

This chapter provides information about the RDM data loads.

Note: The three refresh procedures carry a substantial performance overhead and should be run in parallel. You can do this by setting the variable "Number of parallel processes to run load procedures" to either 4 or 6. This variable can be found in the env.sh file, which is located in the <PL_HOME>/modules/Database/REDWOODSchema/install/oracle/REDWOODSchema/scripts directory.

RDM Initialization Load

The RDM initialization load creates the RDM tables and loads all available historical data. You run the RDM initialization load only once, after you have installed Merchant Desktop (an optional component) and the RDM.

To run the RDM initialization load, run the pl_init_load_rdm.sh script, located in the <PL_HOME>/modules/Database/REDWOODSchema/install/oracle/REDWOODSchema/scripts directory. This script requires the plexports.sh script, which contains variables configured during installation and is located in the <PL_HOME>/config directory.

RDM Initialization Load Procedures

This section describes the all of the RDM initialization load procedures.

The RDM initialization load has the following load-order dependencies:

Table 5–1 RDM Load Procedures

Load Step	Description
1. SetRunStartWeekly	Sets up the weekly update.

Table 5–1 (Cont.) RDM Load Procedures

Load Step	Description
2. LoadPeriods	Refreshes RDM_PERIODS_TBL. This is a complete refresh and is restartable. Populates the RDM_PERIODS_TBL table with all of your historical data.
3a. InitLoadFA	Runs only once, during the RDM initialization load. This procedure is responsible for the initial load of RDM_FORECAST_ACTIVITIES. This is a complete refresh and is restartable.
3b. LoadFA	Refreshes RDM_FORECAST_ACTIVITIES. This is an incremental refresh and is restartable. If this procedure fails, do not truncate or drop RDM_SYSTEM_DB, RDM_TEMP_FA, or RDM_TEMP_FA_UPD. Populates the RDM_FORECAST_ACTIVITIES table with all of your historical data.
4. LoadBudgets	Refreshes RDM_BUDGETS. This is an incremental refresh and is restartable.
5. LoadItemData	Refreshes RDM_ITEM_DATA. This is an incremental refresh and is restartable.
6. RefreshSummarySetup	This procedure must precede RefreshFA and RefreshAct. It is restartable.
7. RefreshFA	Refreshes RDM_MV_FA_n. This is a complete refresh and is restartable.
8. RefreshAct	Refreshes RDM_MV_ACT_n. This is a complete refresh and is restartable.
9. UpdateLoadStatus	This procedure maintains status during a refresh.
10. SetRunEndWeekly	Finishes the weekly update.

RDM Weekly Load

The RDM weekly load updates the RDM tables with data from the model run. Run the RDM weekly load after each model run. The RDM weekly load populates the RDM_FORECAST_ACTIVITIES (LoadFA) and RDM_PERIODS_TBL (LoadPeriods) tables with all of your weekly data.

Running the RDM Weekly Load

To run the RDM weekly load, run the pl_load_rdm.sh script. The script is located in the <PL_HOME>/modules/Database/REDWOODSchema/install/oracle/REDWOODSchema/scripts directory. It requires the plexports.sh script, which contains variables configured during installation and is located in the <PL_HOME>/config directory.

RDM Weekly Load Procedures

This section describes the all of the RDM weekly load procedures.

The RDM weekly load has the following load-order dependencies:

Table 5–2 RDM Weekly Load Procedures

Load Step	Description
1. SetRunStartWeekly	Sets up the weekly update.
2. LoadPeriods	Refreshes RDM_PERIODS_TBL. This is a complete refresh and is restartable.

Table 5–2 (Cont.) RDM Weekly Load Procedures

Load Step	Description
3. LoadFA	Refreshes RDM_FORECAST_ACTIVITIES. This is an incremental refresh and is restartable. If this procedure fails, do not truncate or drop RDM_SYSTEM_DB, RDM_TEMP_FA, or RDM_TEMP_FA_UPD.
4. LoadBudgets	Refreshes RDM_BUDGETS. This is an incremental refresh and is restartable.
5. LoadItemData	Refreshes RDM_ITEM_DATA. This is an incremental refresh and is restartable.
6. RefreshSummarySetup	This procedure must precede RefreshFA and RefreshAct. It is restartable.
7. RefreshFA	Refreshes RDM_MV_FA_n. This is a complete refresh and is restartable.
8. RefreshAct	Refreshes RDM_MV_ACT_n. This is a complete refresh and is restartable.
9. UpdateLoadStatus	This procedure maintains status during a refresh.
10. SetRunEndWeekly	Finishes the weekly update.

RDM Incremental Load

The RDM incremental load updates the RDM item data. Run the incremental load after any markdown activity in Price. The incremental load updates the RDM_ITEM_DATA table.

By default, you should schedule the RDM incremental load to run at least once day, at the end of day. It can also be run more frequently throughout the day, as required by your business practices. You can even run an RDM incremental load even while users are writing reports. However, be aware of potential performance issues.

Running the RDM Incremental Load

To run the RDM incremental load, run the pl_daily_load_rdm.sh script. The script is located in the <PL_HOME>/modules/Database/REDWOODSchema/install/oracle/REDWOODSchema/scripts directory. It requires the plexports.sh script, which contains variables configured during installation and is located in the <PL_HOME>/config directory.

Tuning the RDM Incremental Load Performance

The length of the data load varies, depending on the amount of data being written. A lengthy load may impact the user experience as follows:

- Slow responsiveness. If the incremental load is lengthy, the user interface may appear to hang while generating reports.
- Stale data. When the report finally does completes, it may contain old data from before the load. If the underlying data is in the process of updating while users are trying to generate reports, their reports will contain old data. Note that all data in the report will be at the same state; there is no mix of old and new data.

Avoiding Performance Issues

To avoid potential performance issues, configure the scripts to run as needed for your environment, keeping in mind the following performance guidelines and considerations:

- For the most accurate data, run the RDM incremental load whenever data changes.
- For the best user experience for running reports, avoid running an RDM incremental load when users are online, writing reports.
- Since a load takes approximately 5-15 minutes, schedule loads to occur no more frequently than the time it takes to complete the most time consuming load.
- Updates to the data in the RDM_ITEM_DATA table are particularly resource intensive and take longer to complete.

RDM Incremental Load Procedures

This section describes the RDM incremental load procedures.

The RDM incremental load has the following load-order dependencies:

Table 5–3 RDM Incremental Load Procedures

Load Step	Description
1. SetRunStartDaily	Sets up the incremental update.
2. LoadItemData	Refreshes RDM_ITEM_DATA. This is an incremental refresh and is restartable.
UpdateLoadStatus	This procedure maintains status during a refresh.
4. SetRunEndDaily	Finishes the incremental update.

6

Price Tools

This chapter contains the following:

PriceAdmin Commands

PriceAdmin commands address functions used in model runs and in managing the Price application. These commands can be used in scripts to automate the weekly batch process.

The PriceAdmin commands require the following:

- Java Version
  - For WebLogic, you can use either the Standard Sun JDK Java or the Sun JRockit JRE
- Client-side Library referenced by a command
  - For WebLogic, use **wlclient.jar**
- For p4pgui-based commands (generateSendback, listSendbackTypes, lockWorksheets, unlockWorksheets, disableLogin, and enableLogin) use http with either application server.

You should invoke the PriceAdmin commands from **PriceAdmin.jar** via **com.profitlogic.adm.PriceAdmin**. For example:

A WebLogic run-based command using port 7033

java -cp wlclient.jar:PriceAdmin.jar com.profitlogic.adm.PriceAdmin enableLogin *-s price-server-00:7033*

A WebLogic p4gui-based command

java -cp wlclient.jar:Priceadmin.jar com.profitlogic.adm.Priceadmin listSendbackTypes *-s price-server-00:7033*

The **-v** option directs the logs to **stderr**, which can be redirected to a file.

You can access help on each command by using ? or --help. For example, enter enableLogin -?.

disableLogin

Description: The **disableLogi**n command prevents all users with the exception of the System Administrator from logging into the Price application.

Syntax:

disableLogin *-s <server url> [-v]*

disableLogin *--server <server url> [--verbose]*

Arguments:

-s, --server <server url>	the url of the remote application server
-v, --verbose	displays logging messages as command executes

Return Values: The command returns 0 if it is successful. The command returns a value other than 0 if it fails.

enableLogin

Description: The **enableLogin** command allows all users to log into the Price application.

Syntax:

enableLogin *-s <server url> [-v]*

enableLogin *--server <server url> [--verbose]*

Arguments:

-s, --server <server url>	the url of the remote application server
-v, --verbose	displays logging message as command executes

Return Values: The command returns 0 if it is successful. The command returns a value other than 0 if it fails.

generateAlerts

Description: The **generateAlerts** command generates alerts for all configured alerts of a specified category.

Syntax:

generateAlerts *-c <category of alerts> -m <integer> [-v]*

generateAlerts *--category <category of alerts> --maxThreads <integer> [--verbose]*

Arguments:

-c, --category <category of alerts>	Weekly
-m, --maxThreads <integer>	the number of threads to spawn for generating alerts
-v, --verbose	displays logging messages as command executes

Return Values: The command returns 0 if it is successful. The command returns a value other than 0 if it fails.

generateSendback

Description: The **generateSendback** command creates a file containing data for import to an ERP system.

Syntax:

generateSendback *-t <sendback type> -o <file> -s <server url> [-v]*

generateSendback *--type <sendback type> --output <file> --server <server url> [--verbose]*

Arguments:

-t, -- type<sendback type>	the type of sendback file as provided by the listSendbackTypes command.
-o, --output <filename>	the directory path and filename where the sendback information should be written.
-s, --server <server url>	the url of the remote application server
-v, --verbose	displays logging messages as command executes

Return Values: The command returns 0 if it is successful. The command returns a value other than 0 if it fails.

isSummaryCacheLocked

Description: The **isSummaryCacheLocked** command outputs the status of the summary metrics table.

Syntax:

isSummaryCacheLocked *-s <server url> [-v]*

isSummaryCacheLocked *--server <server url> [--verbose]*

Arguments:

-s, --server <server url>	the url of the remote application server
-v, --verbose	displays logging messages as command executes

Return Values: The command returns 0 if it is successful. The command returns a value other than 0 if it fails.

listSendbackTypes

Description: The **listSendbackTypes** command provides a list of the types of sendback available.

Syntax:

listSendbackTypes *-s <server url> [-v]*

listSendbackTypes *--server <server url> [--verbose]*

Arguments:

-s, --server <server url>	the url of the remote application server
-v, --verbose	displays logging messages as command executes

Return Values: The command returns 0 if it is successful. The command returns a value other than 0 if it fails.

Output: The command provides a list of the available sendback types.

lockWorksheets

Description: The **lockWorksheets** command prevents all access except read-only access to the worksheets.

Syntax:

lockWorksheets *-s <server url> [-v]*

lockWorksheets *--server <server url> [--verbose]*

Arguments:

-s, --server <server url>	the url of the remote application server
-v, --verbose	displays logging messages as command executes

Return Values: The command returns 0 if it is successful. The command returns a value other than 0 if it fails.

refreshForecastCache

Description: The **refreshForecastCache** command refreshes the forecast cache. It operates synchronously. It replaces pinging the WorksheetForecastServlet.

Syntax:

refreshForecastCache *-s <server url> -t <number of threads> [-c]*

refreshForecastCache *--server <server url> --threads <number of threads> [--continue]*

Arguments:

-s, --server <server url>	the url of the remote application server
-t, --threads <number of threads>	the number of threads being used
-c, --continueOnFailure	processing continues on failure even if it fails for some items

Return Values: The command returns 0 if it is successful. The command returns a value other than 0 if it fails.

The correct option to continue on failure is -c or --continueOnFailure. What continue on failure means is that an individual item that's being refreshed could fail, but if that option is passed in all the rest of the items will continue processing. " Continue the process even if it

fails for some items" should be sufficient.

refreshSummaryCache

Description: The **refreshSummaryCache** command causes summary metrics to be recalculated based on current values in the ITEM_DATA table. It should be invoked after every model run or when the configuration changes.

Syntax:

refreshSummaryCache *-s <server url>*

refreshSummaryCache *--server <server url>*

Arguments:

-s, --server <server url>	the url of the remote application server

Return Values: The command returns 0 if it is successful. The command returns a value other than 0 if it fails.

registerAlerts

Description: The **registerAlerts** command registers the Price product and all Price alerts with the Common Alerts Framework.

Syntax:

registerAlerts *-r <register> -s <server url> -u <unregister> [-v]*

registerAlerts *--register <register> --server <server url> --unregister <unregister> [--verbose]*

Arguments:

-r --register <register Price and Price alerts>	registers Price and Price alerts with the Common Alert Framework
-s, --server <server url>	the url of the remote application server

-u --unregister <unregister Price and Price alerts>	unregisters Price and Price alerts with the Common Alert Framework
-v,--verbose	displays logging messages as command executes

Return Values: The command returns 0 if it is successful. The command returns a value other than 0 if it fails.

releaseSummaryCacheLock

Description: The **releaseSummaryCacheLock** command forcibly unlocks the summary metrics table. Cache locking is implemented using a flag that persists in the database, so this command resets the flag to unlocked.

Syntax:

releaseSummaryCacheLock *-s <server url>*

releaseSummaryCacheLock *--server <server url>*

Arguments:

-s, --server <server url>	the url of the remote application server

Return Values: The command returns 0 if it is successful. The command returns a value other than 0 if it fails.

unlockWorksheets

Description: The **unlockWorksheets** command unlocks the worksheets so they are again readable and writable. (See the lockWorksheets command.)

Syntax:

unlockWorksheets *-s <server url> [-v]*

unlockWorksheets *--server <server url> [--verbose]*

Arguments:

-s, --server <server url>	the url of the remote application server
-v,--verbose	displays logging messages as command executes

Return Values: The command returns 0 if it is successful. The command returns a value other than 0 if it fails.

7

Troubleshooting

This chapter contains the following:

Assessing Optimization Run Problems

Problems can occur at any point in the run process. Some errors will prevent the run from completing and others will not. Price provides a report that contains two levels of error messages. this report can be accessed via runReport.sh.

Stoplight Summary

The Stoplight Summary section of the report provides a count of all errors, categorized by the level of severity of the error. The red category indicates system or configuration errors that must be fixed. The yellow category includes errors that result from missing data or a possible mis-configuration of the application. The green category indicates no errors. A run can be considered successful if there are no errors in the red category.

Outcome Messages

The Message Breakdown section of the report lists the errors in the order of severity and provides a count of the number of items affected. The outcome messages are categorized according to the following criteria:

- System errors are caused by conditions that should not occur in a properly configured system. These conditions include system-level problems, application inconsistencies, mis-configurations, communication errors, time-outs, and unrecoverable data failures. Any exception with a severity of Fatal is automatically included here. All System errors are included in the red category of the Stoplight Summary and should be immediately corrected.
- Model Configuration errors indicate that the configuration of the model is incomplete or contains errors/inconsistencies. Information supplied during the configuration, such as optimization parameters or stochastic samples, may be missing, incomplete, or invalid. Errors may also have arisen during the customization of the Inference Rules. All Model Configuration errors are included

in the red category of the Stoplight Summary and should be immediately corrected.

- Item Data errors indicate missing or inaccessible customer data. Either the data is missing, a problem occurred in loading the data, or the data is inaccessible because of a problem with the inference rules.
- Inactivity errors indicate that there is not enough historical information to drive the model as it is currently configured.
- Error indicates problems that are not categorized.
- Markdown Blocked messages indicate that a markdown was not permitted on the effective date because of business rules, promotions, or other factors. A markdown at a later date may have been recommended. This information comes from the inference rules.
- Not Recommended messages indicate that a markdown was possible on the effective date, but it was not recommended. Based on the history, model configuration, and business rules, Price determined that revenue would be maximized by leaving the current price unchanged for at least one more week. This outcome typically occurs because the next markdown price available on the price ladder is lower than the optimal markdown.
- Warning messages indicate a general condition that is less severe than an error and are provided to flag a condition of note, such as a price above the full price.
- Information messages provide general diagnostic information.

Modifying Optimization Run Error Messages

You can change the text of specific error messages that are produced by the Optimization engine during the run.

The default error messages are located in the file, config/suite/resources/EngineResources.properties. You can create a file, also called EngineResources.properties that contains the changed text only. This file should be located in either config/suite/environment/resources or in config/suite/client/resources. Any messages in the EngineResources.properties file in either of these two subdirectories will override the default message text.

Guidelines for Fixing Problems

Here are some guidelines for fixing the problems you have diagnosed.

- Correct system errors first.
- Both Model Configuration errors and Yellow-level errors can cause data problems. If the problem is the result of a mis-configuration, you may want to compare the configuration settings to the original, default settings. When making changes to the configuration, use a systematic, incremental approach and record the effect of any configuration changes.
- Yellow-level messages may indicate data problems or may be legitimate. Data may be missing or invalid. Alternatively, an error may have occurred during the run.
- Note changes in the number of error messages from week to week. Significant changes may indicate a problem.
- When analyzing data problems, consider the following causes:
  - Problems loading the data

- Data missing from database tables
- A mis-configuration of the inference rules that prevents data from being visible
- Business rule parameter changes

Diagnostic Messages

Table 7–1 Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
activityDataNotBuilt Could not build ActivityData for item.	Could not build ActivityData for item.	Error	Data Access layer	Model misconfiguration
alreadyInClearance Already in clearance	A markdown is not possible on the effective date because the item is already in clearance, meaning a clearance markdown has already occurred.	Blocked	Optimization Engine	This sort of clearance is from specific client definition, and depends on how the client is configured.
badCollectionPricing Value Invalid collectionPricing valueInvalid Collection Pricing rule value: <val>	Agorai uses the collectionPricing value to determine the pricing interdependencies when marking down items in the collection. There are three supported values: "MarkdownTogether" (item may be marked down to different prices), "PriceTogether" (items must have same price), or "PercentOffTogether" (items must be the same percentage of the full price).	Model Conf	Optimization Engine	This is likely an error in Didyma customization or data in the business rules table.Should not occur in properly customized system.
badDemandStrategy Invalid Demand Strategy Value:	Agorai's heuristic model uses the demandStrategy provided to it to define how it is to use recent activity data to determine demand. It must be one of the following values: "Average", "Maximum", "TrimmedMean", "TrimmedMinMean".	Model Conf	Optimization Engine	Model configuration error.
badDispatch Error dispatching to Agorai, details: <details>	This message reports problems interfacing with Agorai - unable to load Agorai, bad Agorai input/output, associated exceptions.	System	Data Access Layer	The path to the AgoraiJNI.so file is incorrect in the job.properties file, or the AgoraiJNI.so file is missing or inaccessible, or the AgoraiJNI.so file could not be loaded because the operating system is not configured properly.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
badInWarehouse Count Number of In Warehouse quantities doesn't match size of historical Sales data	The number of historical "in warehouse quantity" reports provided to Agorai is different from the number of historical sales.All the historical arrays passed to Agorai must be the same size.	Model Conf	Optimization Engine	Missing historical data not properly handled by Didyma; error in Didyma customization.Should not occur in properly customized system.
badLastStoreCount Last week of historical activity does not have a good Store Count	The system requires a positive store count for the last week. Historical sales are normalized using store counts, and we need a proper value here to forecast forward.	Model Conf	Optimization Engine	Lack of good recent data, or error in Didyma customization.
badOnHandCount Number of On Hand quantities doesn't match size of historical Sales data	The number of historical "on hand quantity" reports provided to Agorai is different from the number of historical sales.All the historical arrays passed to Agorai must be the same size.	Model Conf	Optimization Engine	Missing historical data not properly handled by Didyma; error in Didyma customization.Should not occur in properly customized system.
badOnOrderCount Number of On Order quantities doesn't match size of historical Sales data	The number of historical "on order quantity" reports provided to Agorai is different from the number of historical sales.All the historical arrays passed to Agorai must be the same size.	Model Conf	Optimization Engine	Missing historical data not properly handled by Didyma; error in Didyma customization. Should not occur in properly customized system.
badPastTicketPrice Count Number of Past Ticket Prices doesn't match size of historical Sales data	The number of historical "past ticket price" reports provided to Agorai is different from the number of historical sales.All the historical arrays passed to Agorai must be the same size.	Model Conf	Optimization Engine	Missing historical data not properly handled by Didyma; error in Didyma customization.Should not occur in properly customized system.
badPriceCount Number of sales different from the number of pricesNumber of Average Sales Prices doesn't match size of historical Sales data	The number of historical "average sales price" reports provided to Agorai is different from the number of sales reports.All the historical arrays passed to Agorai must be the same size.	Model Conf	Optimization Engine	Missing historical data not properly handled by Didyma; error in Didyma customization.Should not occur in properly customized system.
badPriceForHeuristic Price less than or equal to 0 - too small for the Heuristic modelBad Price for heuristic model: <val>	The forecasting engine detected an invalid price.	Model Conf	Optimization Engine	You probably won't see this; it would be caught by an earlier validation.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
badPriceInterp Price Interpretation out of range: <val>	Historical, promo, and price ladder prices provided to Agorai each have an interpretation to indicate the nature of the price: markdown, promo, and so on. This value is out of range.	Model Conf	Optimization Engine	Bug in Didyma or Didyma customization. Bad data not properly screened by Didyma.
badSeasonality Seasonality too small or negative: <val>	A seasonality value provided to Agorai is less than a certain threshold value.	Model Conf	Optimization Engine	This is likely a model misconfiguration.
BadSeasonalityFor Heuristic Seasonality less than or equal to 0 -- too small for Heuristic model.Bad Seasonality value for heuristic model: <val>	The forecasting engine detected an invalid seasonality value.	Model Conf	Optimization Engine	You probably won't see this; it would be caught by an earlier validation.
badStoreCountCount Number of Store Count quantities doesn't match size of historical Sales data	The number of historical "store count" reports provided to Agorai is different from the number of historical sales.All the historical arrays passed to Agorai must be the same size.	Model Conf	Optimization Engine	Missing historical data not properly handled by Didyma; error in Didyma customization.
bayesianDataNotBuilt	Could not build BayesianData for item.	Error	Data Access Layer	Model misconfiguration.
beforeEffectiveDate Effective Date inconsistency	There is inconsistent data within Agorai, where somehow the effective date is considered out of range.	Model Conf	Optimization Engine	This represents an inconsistency of information within Agorai. Most likely grossly bad data or an Agorai defect.
bizPolicyNotBuilt	Could not build BusinessPolicy for item.	Error	Data Access Layer	Model misconfiguration.
businessRuleBlocks Markdown prohibited this week due to Business Rule	A business rule prohibits a markdown this week.	Blocked	Data Access Layer	This is defined in the inference_rules.
clearanceNot Permitted Model not configured for Clearance markdowns	Model is not configured to permit clearance markdowns. Optional "clearance" tags in the request's business policy section trigger this.	Model Conf	Optimization Engine	Could result from an inconsistency in business rules. Probably clearance rule information was provided without a clearance price ladder.
collectionBuild Error building Optimization Request for collection	More information accompanies this to provide detail.	Error	Data Access Layer	Look at the other errors for more info.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
collectionItemErrors Collection contains items with errors	This is put on the collection when errors are found in one or more of its items. It will show for the items without errors as well, to better indicate what occurred.	Error	Optimization Engine	Refer to the item errors to see what occurred.
databaseErrorData	Didyma encountered database connectivity problems while trying to create Agorai input.	Model Conf	Data Access Layer	Configuration/customization error or worse.
dateInvalidDay Date Error - Invalid day: <val>	Problem with the day portion of a date value.	Model Conf	Optimization Engine	Probably bad data from Didyma, perhaps due to data or customization problems.
dateInvalidDayOf Week Date Error - Invalid day of week: <val>	Problem with a day of week value.	Model Conf	Optimization Engine	Probably bad data from Didyma, perhaps due to data or customization problems.
dateInvalidMonth Date Error - Invalid month: <val>	Problem with the month portion of a date value.	Model Conf	Optimization Engine	Probably bad data from Didyma, perhaps due to data or customization problems.
dateInvalidWeek Number Date Error - Invalid week number: <val>	Problem with a week number value.	Model Conf	Optimization Engine	Probably bad data from Didyma, perhaps due to data or customization problems.
dateInvalidYear Date Error - Invalid year: <val>	Problem with the year portion of a date value.	Model Conf	Optimization Engine	Probably bad data from Didyma, perhaps due to data or customization problems.
DistributionDoesNot Sample0First Stochastic Sample distribution does not sample 0 first	The first point of the stochastic sample provided to Agorai must have a multiplier of 0.	Model Conf	Optimization Engine	
duplicateTag ERROR: duplicate tag ({0}) at {1}).	Problem parsing the XML.	Model Conf	Optimization Engine	Didyma probably left something malformed. Perhaps due to data or customization problems, or mixed component versions.
EffectiveDateMust PrecedeOutDate Out Date precedes Effective Date	The out date for this item has already passed or will pass before our recommendations could take effect.	Item Data	Optimization Engine	Can occur normally.An item with an old out date was still selected by the eligibility query (which defines the items/collections to be included in a run). This could be bad data or an error in database configuration.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
effectiveDatesDiffer Effective Dates can't differ within collection	All items in a collection must have the same effective date. In fact, the effective date should be the same for everything in an entire run.	Model Conf	Optimization Engine	Business rule configuration or Didyma customization error.
elementContentError Error in content of XML element (<tag>) at <pos>	Problem parsing the XML. Perhaps a value is the wrong type of missing.	Model Conf	Optimization Engine	Didyma probably left something malformed. Perhaps due to data or customization problems.
emptySeasonalities Empty seasonalitiesNo Seasonality values provided	No seasonality values were provided to Agorai.	Model Conf	Optimization Engine	Improper population of seasonality data in the database or an error in Didyma customization.
errorOpeningLogFile Error opening log file: <file>	Agorai could not open its log file.	System	Optimization Engine	Problem in Agorai's configuration, or a problem in the file system to which logs are directed.
ErrorReadingElement Content Error reading content of XML element (<tag>) at <pos>	Problem parsing the XML. Usually a value is of the wrong type or missing.	Model Conf	Optimization Engine	Didyma provided a value of the wrong type, or it was missing. Perhaps due to data or customization problems.
ErrorWritingElement Content Error writing content of XML element (<tag>)	Agorai is having trouble writing a value in its results. Indicates a major malfunction.	System	Optimization Engine	Probably an Agorai defect.
histPeriodNotPositive Historical Period must be positiveDays Per Historical Interval value is not positive	The daysPerInterval value is not positive. This is the number of days represented by each period of historical data provided to Agorai. It is usually 7 days. Elsewhere where we refer to "weeks of activity" we are usually referring to intervals of this number of days.	Model Conf	Optimization Engine	This is likely an error in Didyma customization or data in the business rules table. Should not occur in properly customized system.
internal The message varies depending on what is found. Many such messages are generated by the operating system and passed through unchanged.	This message reports unexpected internal errors that really should not occur.	System	Optimization Engine	This indicates an unexpected situation. It could be a software bug, or it could be that the software is operating outside of its expected parameters in some way that isn't being rigorously validated. In any case, this situation should be reported to Customer Support.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
invalid Validation error	Appears with other errors that occur during validation. That is, this message groups accumulated validation errors.	Error	Optimization Engine	See the associated messages for the details.
InvalidClearanceSell through Invalid Clearance Sell-Through rule value: <val>	The clearance sellthrough value provided to Agorai must be between 0 and 1. This is the optional sell-through ratio that triggers a clearance markdown.	Model Conf	Optimization Engine	Business rule configuration or Didyma customization error.
invalidClearanceLock out Invalid Clearance Lockout rule value: <val>	The clearance lockout value provided to Agorai must be >= 0. This is the optional number of days before the out date where a clearance markdown is forced.	Model Conf	Optimization Engine	Business rule configuration or Didyma customization error.
InvalidCumulative TotalUnitsSold Invalid cumulative total units soldSales data contains invalid cumulative total units sold	SalesData contains an invalid total unit sold value.	Item Data	Optimization Engine	Likely a model configuration error.
invalidDemand Multiplier Invalid demand multiplier. Check stochastic demand uncertainty and sample multipliers.	Calculation of the demand multiplier involves the product of the demand uncertainty and a stochastic sample's multiplier value. In this case it yielded an number <=0.	Model Conf	Optimization Engine	Most likely a misconfiguration of the model.Should not occur in properly customized system.
invalidInterpInPrice Ladder Price Interpretation not appropriate for Price Ladder: <val>	A price ladder supplied to Agorai contains at least one price with an interpretation other than 1 (markdown start) or 6 (clearance start).	Model Conf	Optimization Engine	Likely a model configuration error.
InvalidInterpretation ForPromo Invalid Price Interpretation for Promo: <val>	Promo information provided to Agorai contains prices without a promo interpretation (2, 3, or 9.)	Model Conf	Optimization Engine	Likely a model configuration error.
invalidInventory Target Invalid Target Inventory value: <val>	The inventory target value provided to Agorai must be >= 0. This is the desired inventory at the target date.	Model Conf	Optimization Engine	Business rule configuration or model configuration error.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
invalidMaxAbsolute Price Invalid Maximum Absolute Markdown Price: <val>	The max absolute price value provided to Agorai must be between 0 and 1.A markdown recommendation may not yield a price higher than this. A price relative to ticketPrice.	Model Conf	Optimization Engine	Business rule configuration or model configuration error.
InvalidMaxFirstMarkdownPercent Invalid Maximum First Markdown Percentage: <val>	The maximum first markdown percentage value provided to Agorai must be between 0 and 1.This is the maximum markdown percentage (0 to 1) for the first markdown, as compared to the current price.	Model Conf	Optimization Engine	Business rule configuration or model configuration error.
InvalidMaxNumberOfMarkdowns Invalid Maximum Number of Markdowns: <val>	The max number of markdowns value provided to Agorai must be > 0.Maximum number of markdowns permitted during item's entire life cycle. This includes markdowns that already occurred in history.	Model Conf	Optimization Engine	Business rule configuration or model configuration error.
invalidMaxPrice Invalid Maximum Absolute Markdown Price: <val>	Value supplied for maxAbsolutePrice was not from 0 to 1.A markdown recommendation may not yield a price higher than this. A price relative to ticketPrice.	Model Conf	Optimization Engine	Business rule configuration or model configuration error.
InvalidMaxSubsequentMarkdownPercent Invalid Maximum Subsequent Markdown Percentage: <val>	The maximum subsequent markdown percentage value provided to Agorai must be between 0 and 1.Maximum markdown percentage (0 to 1) for all markdowns after the first, as compared to the current price.	Model Conf	Optimization Engine	Business rule configuration or model configuration error.
InvalidMinFirstMarkdownPercent Invalid Minimum First Markdown Percentage: <val>	The minimum first markdown percentage value provided to Agorai must be between 0 and 1.This is the minimum markdown percentage (0 to 1) allowed for the first markdown, as compared to the current price.	Model Conf	Optimization Engine	Business rule configuration or model configuration error.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
InvalidMinMarkdown Interval Invalid Min Markdown Interval value: <val>	The minMarkdownInterval value provided to Agorai was <= 0.	Model Conf	Optimization Engine	
InvalidMinMarkdown Intervals Invalid dated Min Markdown Interval value: <val>	A (date-specific) minMarkdownIntervals value provided to Agorai was <= 0.	Model Conf	Optimization Engine	
InvalidMinmarkdown IntervalsDates Invalid dates for Min Markdown Intervals: <start> > <end>	A improper date range was supplied in minMarkdownIntervals.	Model Conf	Optimization Engine	
InvalidMinMarkdown PercentOfFullPrice Invalid Minimum Markdown Percentage of Full Price: <val>	The minimum markdown percentage of full price value provided to Agorai must be between 0 and 1.Minimum step of each markdown as a percentage (0-1) of the full price.	Model Conf	Optimization Engine	Business rule configuration or model configuration error.
InvalidMinSubsequentMarkdown Percent Invalid Minimum Subsequent Markdown Percentage: <val>	The minimum subsequent markdown percentage value provided to Agorai must be between 0 and 1.Minimum markdown percentage (0 to 1) for all markdowns after the first, as compared to the current price.	Model Conf	Optimization Engine	Business rule configuration or model configuration error.
InvalidNumberOf UnitsOnHand Sales data contains invalid number of units on hand: <val>	SalesData contains an invalid unit on hand value.	Item Data	Optimization Engine	Didyma customization error or Agorai defect.
invalidObjectType Invalid Object Type: <val>	Problem identifying objects for status messages.	System	Optimization Engine	Probably an internal problem.
invalidPriceInterpretation Price Interpretation out of range: <val>	Same as badPriceInterp, but in slightly different code.Historical, promo, and price ladder prices provided to Agorai each have an interpretation to indicate the nature of the price: markdown, promo, and so on. This value is out of range.	Model Conf	Optimization Engine	Defect in model configuration.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
invalidRelativePrice Invalid relative price	A price is less than or equal to zero and not marked as "Bad data".	Model Conf	Optimization Engine	Be suspicious.Bad price data or error in model configuration.
invalidSalesInterpretation Sales data contains invalid Interpretation: <val>	SalesData contains an invalid interpretation value.	Model Conf	Optimization Engine	Model configuration error.
invalidSeasonalityLength Improper number of Seasonality values provided: <val> days	Agorai needs to be supplied with either 52 or 53 weeks of seasonality. This is by interface contract.	Model Conf	Optimization Engine	Seasonality data or model configuration, most likely the latter.
invalidSeverity Invalid Severity: <val>	Problem identifying severity (status level) for status messages.	System	Optimization Engine	Probably an internal problem. Should be reported to Customer Support.
invalidStoreCount Sales data contains invalid store count: <val>	SalesData contains an invalid store count value.	Item Data	Optimization Engine	Model configuration error.
invalidTicketPrice Invalid Full Price: <val>	The ticket price (full price) provided to Agorai must be >= 0.	Item Data	Optimization Engine	
invalidTotalDollarSales Sales data contains invalid total-dollar sales: <val>	SalesData contains an invalid total dollar sales value.	Item Data	Optimization Engine	Model configuration error.
itemBlocksMarkdown Item blocks markdown for collection on Effective Date	At least one item in the collection does not permit a markdown on the effective date, therefore the collection as a whole can not be marked down.	Blocked	Optimization Engine	Look at the associated item messages for more information.
itemBuild Error building Optimization Request for item	More information accompanies this to provide detail.	Error	Data Access Layer	Look at the other errors for more info.
LastSalesDataDiffersFromStartSimulation End of Sales data <date> differs from Simulation Start Date <date>	Inconsistency in historical data.	Model Conf	Optimization Engine	Model configuration error.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
MarkdownTogether Advanced Item recommended this week due to MarkdownTogether collection	This item in a "markdown together" collection would not have been marked down this week by itself, but is recommended because of the collection.	Warning	Optimization Engine	Can occur in normal situations.
MarkdownTogether Delayed Item not recommended for this week due to MarkdownTogether collection	This item in a "markdown together" collection would have been marked down, but was blocked due to the collection.	Not Rec	Optimization Engine	Can occur in normal situations
missingMarkdown Dates	Item is missing markdown dates.	Error	Data Access Layer	Markdown date missing (not start date or out date; those are caught individually).
missingTag XML error: Missing tag (<tag>) at <pos>	Problem parsing the XML.	Model Conf	Optimization Engine	Didyma probably left something malformed. Perhaps due to data or customization problems, or mixed component versions.
missingTagErrors XML error: Missing tags	Problem parsing the XML. This collects a number of missing tag errors.	Model Conf	Optimization Engine	Didyma probably left something malformed. Perhaps due to data or customization problems, or mixed component versions.
needMoreSales Not enough historical Sales data to determine demand	This occurs when the sales history, as trimmed according to the configured demandStrategy, has fewer entries than needed, as determined by demandIntervals. (These are both configurable).	Inactivity	Optimization Engine	Can occur normally.Most likely this is sales inactivity, although it is possible the demandStrategy and/or demandIntervals are misconfigured.
NoAdditionalMarkdownsPermitted No additional markdowns permitted	maxNumberMarkdowns has already been reached for this item.	Blocked	Optimization Engine	Can occur in normal situations.
noCandidatePrices No candidate prices available for markdown: filtered Price Ladder is empty	The price ladder coming in to Agorai is empty. Didyma has filtered the price ladder so it contains only values below the current price and valid according to various business rules, in this case there is nothing left.	Item Data	Optimization Engine	Lack of good recent activity data, or error is Price Ladder data or configuration.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
noCollectionPricing OrID	Collection is missing collection pricing or ID.	Error	Data Access Layer	Model misconfiguration.
noDataForForecast No historical Sales and Price data available to determine demand	This occurs when Agorai goes through the sales history to get the normalized demand and does not find any entries that have both "known" (not "unavailable") sales data and "valid" (not unknown) price data (average selling price).	Inactivity	Optimization Engine	Can occur normally.
noFullPrice This item has no Full Price	Didyma was unable to obtain a full price for an item.	Item Data	Data Access Layer	Some items without a full price are getting though the eligibility query. It could also be an error in Didyma customization.
noGamma This item missing Price Elasticity (gamma)	Didyma was unable to obtain a price elasticity value (gamma) for this item.	Model Conf	Data Access Layer	Model misconfiguration.
noInventory There is no inventory to clear	The last week of historical activity provided to activity has no units remaining.	Item Data	Optimization Engine	
noItemsFoundFor Collection	No items found for this collection.	Error	Data Access Layer	Data configuration not consistent.
noItemsInCollection No Items in this Collection	No items in a collection. This represents a collection-mapping problem prior to Agorai.	Model Conf	Optimization Engine	Should never have been sent to Agorai. Likely a customization problem.
NoMarkdownPrices AtEffectiveDate No markdown prices available on Effective Date	There were no legal price candidates to consider for markdowns at the effective date.	Blocked	Optimization Engine	Look at current price, price ladders, business rule price restrictions.
noOutDate This item has no Out Date	Didyma was unable to obtain an out date for an item.	Item Data	Data Access Layer	Some items without an out date are getting though the eligibility query. It could also be an error in model configuration.
noPriceAtEffective Date No price is available on Effective Date	The model was unable to determine the price on the effective date.	Blocked	Optimization Engine	Something occurred in the simulation before the effective date that made this information unavailable.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
noSampleDistribution No Stochastic Sample distribution provided	There must be at least one point in the stochastic sample supplied to Agorai.	Model Conf	Optimization Engine	
noStartDate This item has no Start Date	Didyma was unable to obtain a start date for an item.	Item Data	Data Access Layer	Some items without a start date are getting though the eligibility query. It could also be an error in model configuration.
NotEnoughFor TrimmedMean Not enough clean Sales history to take a Trimmed Mean	This error occurs because the system got some normalized sales history, but there are fewer entries than required. This problem occurs only occur with the TrimmedMean demandStrategy.	Inactivity	Optimization Engine	Lack of good recent activity data., or demandIntervals/demand Strategy misconfiguration.
notEnoughForMean Not enough data to take a mean.Not enough clean Sales history to take an Average	This error occurs because the system got some normalized sales history, but there are fewer entries than required. This problem occurs only occur with the Average demandStrategy.	Inactivity	Optimization Engine	Lack of good recent activity data., or demandIntervals/demand Strategy misconfiguration.
NotEnoughForTrimmedMinMean Not enough clean Sales history to take a Trimmed Min Mean	This error occurs because the system got some normalized sales history, but there are fewer entries than required. This problem occurs only occur with the TrimmedMinMean demandStrategy.	Inactivity	Optimization Engine	Lack of good recent activity data., or demanIntervals/demand Strategy misconfiguration.
noTicketPrice This item has no Ticket Price	Didyma was unable to obtain a ticket price for an item. (ticketPrice tag in request)	Item Data	Data Access Layer	Some items without a ticket price are getting though the eligibility query. It could also be an error in model configuration.
notInClearance Calendar Clearance markdown not permitted by Clearance Markdown Calendar	Business rules are attempting to force a rule-based clearance markdown, but the date is not available in the clearance calendar.	Blocked	Optimization Engine	

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
notInMarkdownCalendar Markdown not permitted by Markdown Calendar	The effective date was not included in the calendar of permitted markdown dates provided to Agorai. The date may not be in the client's markdown calendar, or may have been filtered out by Didyma due to a promo or business rule.	Blocked	Optimization Engine	Messages emitted by Didyma may provide more information.
notRecommended Markdown not recommended for this week	A markdown was possible this week, but was not deemed optimal.	Not Rec	Optimization Engine	This usually means that the price is set correctly, which is what you were hoping for! But if the item is one that "should" have gotten a markdown, then you might want to check into why it did not.
noUsefulSales Data is too dirty to deal with -- zero useful sales foundData too dirty to determine demand: zero useful sales found	Applying the demandStrategy to determine normalized sales yielded a value <= 0	Inactivity	Optimization Engine	Activity Data values are not adequate for the heuristic model as configured.
outDateTooFarAhead Builder Error: outDate must be within 560 days of lastDataDate Out Date cannot be more than <n> days after the Start of Simulation	This error prevents wasting time optimizing items whose out date is wildly in the future. The error occurs when the Out Date is more than 560 days after the Start of the Simulation (first forecasted day). Specifically, this refers to the outDate and startSimulationDate values provided to Agorai in markdownDates.	Item Data	Optimization Engine	Could occur normally, but is cause for suspicion.Could be missing activity data or an erroneous out date. Or an error in Didyma customization yielding same.
pastOutDate Already past Out Date	A markdown is not possible on the effective date because the out date will have already passed.	Blocked	Optimization Engine	Can occur normally.
pastOutOfStockDate Already past Out Date	A markdown is not possible on the effective date because the out date will have already passed.	Blocked	Optimization Engine	Can occur normally

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
priceAboveFull A relative price is higher than the full price	This is a warning that occurs when a relative price is greater than 1, meaning it is higher than the full price.	Warning	Optimization Engine	Be suspicious.Relative prices are supplied to Agorai in many contexts: Average Sales Prices, Past Ticket Prices, Price Ladders, and Planned Promotions, This may be intrinsic to the client's data or an error in customization.
promoBlocks Markdown prohibited this week due to Promotion	A promotion prohibits a markdown this week.	Blocked	Data Access Layer	This is defined in the inference_rules.
promoStartAfterEnd Promo Start Date later than End Date: <start> > <end>	Bad promo data provided to Agorai.	Item Data	Optimization Engine	Bad data or error in model configuration.
SalesDataStartDiffers FromStartDate Date of first Sales data <date> differs from Start Date <date>	Inconsistency in historical data.	Model Conf	Optimization Engine	Bad data or error in model configuration.
sellsOutWithout Changes Sells to target without changes; Markdown not recommended	This simply means that this item or the collection it is in sells out to the inventory target without any new markdowns. Given this, we do not attempt any recommendations as they are deemed to be of limited utility.	Not Rec	Optimization Engine	This was introduced in 3.0 to avoid recommending markdowns that appear to be superfluous and have limited value, usually in low inventory situations.
StartDateMustPrecede SimulationStart Builder Error: startDate must precede lastSalesDataStart of Simulation cannot be before Start of historical data	These two dates provided to Agorai do not make sense together. How can the start of our historic sales activity be after the end of it?The Start Date (beginning of history provided to Agorai) is not before the Start Simulation Date (first forecasted day).	Model Conf	Optimization Engine	Bad data or model configuration error.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
StartSimulationDate MustPrecedeEffective Date Builder Error: lastSalesData must precede effectiveDate Simulation Start Date cannot be after the Effective Date	These two dates provided to Agorai do not make sense together. That is, how can we have activity reports after the date our recommendations are to take effect?Specifically, inconsistent startSimulationDate and effectiveDate values have been provided to Agorai in the markdownDates section.	Item Data	Optimization Engine	Bad data or model configuration error.
stochasticDataNot Built Could not build StochasticData for item.	The data required to build a necessary forecasting feature could not be read from the database.	Error	Data Access Layer	Model misconfiguration.
ticketPriceTooLow ticket price too lowFull Price too low: <val>	The ticket price (full price) provided to Agorai in the Activity Data is less than 0.005.	Item Data	Optimization Engine	Bad data or error in Didyma customization.
tooFewSales No historical Sales data provided	There were no historical sales provided to Agorai.	Inactivity	Optimization Engine	Can occur normally.
tooManymarkdowns No additional markdowns permitted	A markdown is not permitted on the effective date because maxNumberMarkdowns has already been reached for this item.	Blocked	Optimization Engine	Can occur normally.
tooManyValidation Errors Too many validation errors	Some validation errors were suppressed because there were too many of them.	Error	Optimization Engine	Look at the other messages.
tooSoonForMark down Too soon for new markdown	The minimum markdown interval has not passed since the last markdown. See minMarkdownInterval and dated minMarkdownIntervals.	Blocked	Optimization Engine	Can occur normally.
topLevelData	This item is missing some item parameters.	Error	Data Access Layer	Something missing from ir_item_parameters.
unexpectedTag XML error: Unexpected tag (<tag>) at <pos>. Expected tag (<tag>)	Problem parsing the XML.	Model Conf	Optimization Engine	Didyma probably left something malformed. Perhaps due to data or customization problems, or mixed component versions.

Table 7–1 (Cont.) Price Diagnostic Messages

Message ID/ Message Text	Description	Outcome Category	Source	Possible Cause/Solutions
unknown_resource_ key Unknown i18n Engine Resource Key	Indicates no resource value was found for a resource key. For example, a message was generated by resource key in the Engine, but no text for it was defined in the engine resource bundle.	Model Configuration	System	May indicate a version skew with the resource.properties files, or an error is customization or localization of those files.
UnknownError ReadingElement Content Unknown error reading content of XML element (<tag>) at <pos>	Problem parsing the XML. Perhaps a value is of the wrong type or missing.	Model Conf	Optimization Engine	Didyma probably left something malformed. Perhaps due to data or customization problems.
unknownTag XML error: unknown tag (<tag>) at <pos>.	Problem parsing the XML.	Model Conf	Optimization Engine	Didyma probably left something malformed. Perhaps due to data or customization problems, or mixed component versions.
version Agorai version: <buildid>	Shows the build version number of the Optimization Engine.	Informational	Optimization Engine	Not an error; for diagnostic purposes.
version Didyma version: <build>	Shows the build version number of the Data Access Layer	Informational	Data Access Layer	Not an error; for diagnostic purposes.
writeResults Didyma error writing results	This groups other errors that occurred during the Calc Engine's Write Optimization Results step.	Error	Data Access Layer	See associated messages for more info.

FAQs

This section provides answers to frequently asked questions.

Two levels of solution are provided for each question:

- Verification can be done by anyone with the appropriate level of access to the Price application and its components as a way to define the scope of the problem.
- Level I support is the responsibility of someone with SQL knowledge who has access to the database tables, front end configuration files, and log files.

If additional support is required to address a problem, contact Oracle Customer Support.

How do I know what these error messages in the Item Details pop-up mean?

Price requires many pieces of information to generate a forecast for any particular item. Key data include:

- Item Definition (Merchandise Hierarchy, Location Hierarchy)
- Sales

- Business Rules
- Item Life cycle (Dates, Analytical Parameters)

If any of this data is missing, the model will not be able to generate a forecast and the user may notice an error message in the Item Details pop-up.

Verification.

- Check the item for the exact error message.
- Check the Price User Guide to learn more about the error message.
- Common conditions include:
  - Missing start or exit date
  - Missing sales data
  - Missing analytical information
  - Missing price ladder information
  - No inventory to clear
  - Item is past exit date
  - Bad price information (e.g., $0, missing full price, missing current price)
  - No eligible date left in markdown calendar

Level 1.

Most items that do not receive forecasts are not currently on sale. For non-forecasted items that are actually being sold, client data may be incorrect. Relevant database information can be found as follows:

- Start or exit dates in ITEMS view
- Sales or inventory data in ACTIVITIES
- Items in ITEMS_TBL
- Location in LOCATION_HIERARCHY_TBL
- Merchandise in MERCHANDISE_HIERARCHY_TBL and MERCHANDISE_TBL
- Price ladder information in PRICE_LADDERS_TBL and PRICE_LADDER_VALUES_TBL
- Prices in ACTIVITIES and the ITEMS view
- Markdown calendar in IR_MARKDOWN_CALENDAR view
- Business rules in IR_BUSINESS_POLICY view
- Front end information in ITEM_DATA

I do not see a specific item in the application. Where is it?

If data is missing, the item will not be displayed in the front end grids. This is because the eligibility criteria, which set the conditions necessary for an item to go through the optimization run and be available in Markdowns or Maintaining Merchandise, are configured to filter out old or non-selling items.

Verification.

Do the following:

- Try to locate the item in the Markdowns section of the appropriate worksheet.

- Try to locate the item in Maintaining Merchandise, as the item may not have a recommendation.
- Try to locate the item in the Business Rule Manager. The item will appear in the BRM but not in Markdowns or Maintaining Merchandise if it fails to meet the eligibility criteria.

Level 1.

Do the following:

- Try to locate the item in ITEM_DATA. If the item is in the ITEM_DATA table, it should be available in Markdowns or Maintaining Merchandise.
- Try to locate the item in MERCHANDISE_TBL. The item may be in a different part of the hierarchy than expected.
- The item may be in more than one place. If an item is re-classified but not re-named, then the new item will be created in the new merchandise hierarchy without the associated sales and price history. Such an item fails eligibility criteria.
- Check other eligibility components, including sales or store location thresholds, start date/first receipt date, and exit date.

Why is the value for this metric incorrect?

The Price UI is highly configurable and is implemented to a client's specifications. Each implementation includes configured metrics for the Worksheet, Worksheet Summary, and Maintaining Merchandise grids. What If is not configurable.

Verification.

Do the following:

- Verify the metric calculation in the requirements document.
- Metrics can vary, depending on the specific conditions.
- Metrics can aggregate in a variety of ways.
- Metrics can require values such as forecasts, recommendations, and markdowns before they populate, so verify that any conditions have been met.

Level 1.

Do the following:

- Verify the derivation of the metric using the front end configuration files. A Metric either comes from ITEM_DATA or is a calculation based on columns in ITEM_ DATA. Logic should match requirements specifications.
- Check for logic that handles missing data or for easily confused values such as Perm Ticket Price/Current Ticket Price.
- Verify that the underlying ITEM_DATA values contained what is expected.

A What If scenario does not make sense. What is happening?

What If is designed to provide information about an Item forecast or a Group forecast to the end-of-life. Most What If questions are actually forecast questions, such as"Why does an item's forecast show the incorrect sell through?"

Verification.

Do the following:

- Check the item's Sell Through % and other settings in the BRM. Check the EOL Unit Inv and EOL Unit Sales in What If. The model is always constrained by the business rules, even if the result is that an item has inventory left over.
- Make sure the item is not already at its lowest possible price.

Level 1.

Do the following:

- Review the forecast and check to see if the model is projecting any markdowns. The absence of price changes usually means that the model is constrained from making a recommendation. Possible reasons for this include:
  - Business rule conflicts such as Min MD % Off > Max MD % Off.
  - The Max MD # has been reached.
  - The markdown calendar is too sparse.
  - The price ladder is too sparse.
- It the item does not achieve its Sell Thru target with any valid pricing strategy, the model optimizes Gross Margin using a lower Sell Thru value.
- All business rules always apply, even if this means there are no possible prices left for the model to recommend. Business rule conflicts can be especially complicated in Price Group scenarios.
- When researching forecasting, take price ladder information from IR_PRICE_ LADDER. Logic may be built into this view that is used to trim price ladders in order to implement complex rules.
- Markdowns are usually blocked during the last week or two of life. This could be an issue in implementations with sparse markdown calendars.
- The Min MD % Off and Max MD % Off business rules often interact poorly with sparse price ladders. This can cause problems when items have been marked down outside the system in implementations that do not use the Markdowns Taken data feed. Items remain bound by 1st Markdown rules (which can be stricter) even though the user is expecting the item to be impacted by the Subsequent Markdown rules.

Why is my Pricing Group recommendation different than the Item recommendation?

Pricing Groups provide another forecasting option by allowing an item to be forecasted along with similar items (such as other colors in the same style). Pricing groups can be powerful when used appropriately, such as at the beginning of life for a set of items that have the same expected life cycle. Confusing results can occur if pricing groups are not managed correctly.

Verification.

Do the following:

- Check the item recommendation and the pricing group recommendation.
- Check the EOL metrics for the item and the group, if they are available. If both the item and the pricing group hit Sell Thru, then there is usually not a problem. Since a pricing group introduces an additional constraint to a scenario, pricing group results may be less optimal than item results. The optimal path may be different for pricing groups and items.

Level 1.

Do the following:

- Use ITEM_DATA and FORECAST_ACTIVITIES to check item and pricing group forecasts. What If only displays one of the forecasts, and both together can provide additional context.
- A pricing group receives a recommendation for all forecasted items in the same week. However, item recommendations can be staggered or at different depths. For example, if the model expects to make recommendations for two items (for $14.99 and $11.99, respectively) in a group in the following week, it may still make sense to make a recommendation for the group as a whole for $13.99 for the current week.
- A shallow group recommendation and a deep item recommendation in the current week may be offset by a deeper group recommendation and a shallower item recommendation at a later date.
- Check business rules and item prices. Business rule conflicts are more common in Pricing Group scenarios. Business rules must be satisfied for all forecasted items in the group in order for a recommendation to be possible.
- Pricing groups that contain items at different prices can be caught between Min and Max rules (for example, an item at $20.00 with a 50% Max MD and an item at $12.99 with a 15% Min MD).
- Items with different full prices may have different minimum prices. Pricing groups cannot project recommendations past the exit date of any forecasted item. However, once the item is past its exit date, it is dropped from the group forecast.
- Check the item history in ITEM_DATA and HIST_MARKDOWNS_TBL. Items in the No Touch period will prevent the group from receiving a recommendation.
- If the Take Advanced Markdown is used on an individual item within a group, then, in subsequent weeks, other items in the group can receive recommendations, but the group as a whole will be blocked from receiving recommendations.

Why doesn't the forecast take my promotion into consideration?

Bad promotion data is a frequent cause of failure. Common problems with promotion data include duplicate promotions at different price points, end dates that occur before start dates, and promotions that occur in the past. From an analytical perspective, promotions can be a source of forecast inaccuracy. If a promotion is created at the last minute, the model may not have time to respond. For some items, the promotional effect can be difficult to evaluate.

Verification.

Do the following:

- Verify that the promotion metrics are populated in the worksheet.
- Verify the promotion information in What If. The promo pop-up should be available and populated. The sales lift should be evident during the promotion period.
- If promotions block markdown recommendations, make sure that the model is correctly timing markdowns.

Level 1.

Do the following:

- Check the PLANNED_PROMOS_TBL to determine if the promotions data has been loaded correctly.

- Promotions may be entered at higher levels of the Merchandise Hierarchy. PLANNED_PROMO_MAPS_TBL maps individual items to promotions.
- Items may have multiple promotion periods and different concurrent promotion types.
- Check ITEM_DATA to determine if the promotion information has been populated. Information includes Lowest_Future_Promote_Price and Lowest_Future_Promo_Date.
- Check FORECAST_ACTIVITIES to make sure that promotions are being considered. Sales_Price should be the promotion price for full week promotions. Sales_price should be between ticket price and the promotion price for partial-week promotions.

Why is the report metric showing X instead of Y?

Reports can provide further insight into Price data.

Verification.

Do the following:

- Check the worksheet in Price and validate the information there against the information in the report. Make sure the data has not changed between the time the worksheet was modified and the time the report was run.
- Check the report requirements. The report may have been designed to show a different view of the data than the worksheet.

Level 1.

Do the following:

- Standard reports are configured in the same way as the UI, so the same considerations apply.
- Common mistakes include inconsistencies between UI derivations and report derivations and aggregation errors.
- Check report templates that use macros or pivot tables.

Did I change the settings in the BRM correctly?

The BRM is used to manage customer-defined business rules at defined levels. Any setting mistakes can impact Price recommendations.

Verification.

Do the following:

- Make sure the entered values are correct and are for the correct attribute.
- Make sure the entered values are at the correct level in the Merchandise Hierarchy and Location Hierarchy.

Level 1.

Do the following:

- Query BRM_INSTANCE_TBL for the settings at all levels for a given attribute.
- Evaluate the actual results as soon as possible after the next model run.

Why are the No Touch periods behaving in an unexpected manner?

The No Touch rules govern the amount of time at the beginning of life during which an item is not eligible for a recommendation. This is used so that an item can produce a reasonable sales trend, so that the model does not mis-interpret a partial week of sales or a staggered flow pattern and produce poor recommendations.

No Touch rules also govern the amount of time between a markdown taking effect and an item being eligible for another recommendation.

Verification.

Do the following:

- Check the markdown number in the application. An item may not have received a markdown.
- If the locations for a single item show different markdown numbers, then the item may have been taken in some locations and is currently being recommended in others.
- Make sure the markdown's requirements are valid.
- Check the timing of previous markdown actions.

Level 1.

Do the following:

- Check HIST_MARKDOWN_TBL. If no entries exist for the most recent week, the sendback may not have functioned correctly.
- If no approved markdowns exist for the most recent week for a specific worksheet, then the worksheet may not have been approved before the cutoff time.
- If approved markdowns exist for a specific worksheet but not for particular items, then those items were probably not taken.

My implementation of Price includes a custom weekly process, which seems to have failed. How can I confirm this?

Client-specific custom logic is configured using tables, view, and procedures that have a prefix of ISC_.

For example, *forcing* is used to highlight items that have not been taken to markdown and have not received recommendations. These items are added to the worksheet. The status metric is populated to indicate the action. A price point may be chosen. A custom rule could be created that specified that if an item is three weeks from its exit date and only on its second markdown, for it to 75% off. The following steps are recommended if the item is not forced.

Verification.

Do the following:

- Check the logic of the Forcing logic for conditions that include time to exit date, specific cutoff dates, inventory threshold, and markdown number.

Level 1.

Do the following:

- Check ITEM_DATA to determine if the forcing requirements have been met.
- The status flag is usually configured in User_Text_*.

- Non_Recommend_Add_To_Sheet must be populated for an item to appear on the screen.
- A status flag setting without Non_Recommend_Add_To_Sheet may indicate that the item has been removed via Edit Worksheet.

Some Metrics Troubleshooting Tips

Here are solutions to some common metrics issues.

String Calculations

NULL has a type of character. All the other fields in a derivation are numbers. Price can crash when trying to perform a calculation. To avoid this situation, convert NULL from a character to a number type.

Example of Issue:

```
(DECODE(NVL(EOL_CUM_UNIT_SALES,0) + NVL(ENDING_INVENTORY_UNITS,0),  0, NULL,
NVL(EOL_CUM_UNIT_SALES,0) / (NVL(EOL_CUM_UNIT_SALES,0) + NVL(ENDING_INVENTORY_
UNITS,0))))
```

Example of Fix:

```
(DECODE(NVL(EOL_CUM_UNIT_SALES,0) + NVL(ENDING_INVENTORY_UNITS,0),  0, TO_
NUMBER(NULL),  NVL(EOL_CUM_UNIT_SALES,0) / (NVL(EOL_CUM_UNIT_SALES,0) +
NVL(ENDING_INVENTORY_UNITS,0))))
```

Checking for NULL in xml Derivations

When a metric or a where clause in a report is configured, check the use of NULL.

Example of Issue:

CURRENT_UNITS_ON_HAND + CURRENT_UNITS_ON_ORDER

This will return NULL if one of the fields are NULL/blank.

Example of Fix:

(case when CURRENT_UNITS_ON_HAND is null then 0 end) + (case when CURRENT_UNITS_ON_ORDER is null then 0 end)

This will return at least one value, if the other field is NULL/blank.

Division in an xml Derivation

Division by zero should be avoided.

Example of Issue:

1-CURRENT_RETAIL_PRICE / ORIGINAL_RETAIL_PRICE

This does not avoid a division by 0.

Example of fix:

case when ORIGINAL_RETAIL_PRICE <> 0 then 1-CURRENT_RETAIL_PRICE/ORIGINAL_RETAIL_PRICE end

This ensures there is no division by 0.

Weighted Averages in Reports

The weighted average must be handled differently in reports than in regular xml configuration files.

The weighted average in the reports differs from that in regular XML configuration files. In the regular application files, you need to specify P4P_AVG and the argument. In the reports, you need to take additional steps.Basically, the reports use P4P_DIVIDE function, which takes two arguments: numerator and denominator. For example, on the aggregated level, the weighted average for the_price is calculated as sum (the_price*the_inventory)/sum (the_inventory). (By default, it is sum(...)/sum(..), but it could use other types of aggregations, such as sum(...)/max(..). If you calculate a weighted average for a price, you should use the same type, or amount, of inventory in the numerator and denominator. For example, if you need to specify the taken markdown price on the aggregated level, it should be calculated as sum (taken_price*inventory)/sum (taken inventory). Note that since this is the taken price, the denominator must also have the sum of inventories for the taken items only, not sum of all inventories.

So, in this example you should do following:

Create a custom column with the derivation="case when markdown_flag='y' then proposed_price * inventory_units end". Let's assume it has rptTakenPriceTimesInv as a key. Most probably, this is a hidden column. Create another custom column with the derivation="case when markdown_flag='y' then inventory_units end". Let's assume it has rptTakenInventory as a key. Most probably, this is also a hidden column, unless you need to specify it explicitly in the report. Specify the newly created columns in your report. On the aggregated level, your taken price metric will have P4P_DIVIDE function with rptTakenPriceTimesInv as the first argument and rptTakenInventory as the second argument.