ProfitLogic

ProfitLogic Price 4.5
Operations Guide

Contents

Preface

1. Introduction

About the Price Operations Guide 2
What's In This Book 3

2. Standard Interface

Introduction 7
Price Standard Interface Descriptions 8
Merchandise Hierarchy Standard Interface 9
How Price Uses the Merchandise Hierarchy Data 10
Data Fields 10
An Example 11
Technical Notes 11
Location Hierarchy Standard Interface 12
How Price Uses the Location Hierarchy Data 12
Data Fields 12
An Example 13
Technical Notes 14
Calendar Standard Interface 14
How Price Uses the Calendar Data 15
Data Fields 15
An Example 15
Technical Notes 16
Items Standard Interface 16
How Price Uses the Items Data 16
Data Fields 16
An Example 17
Technical Notes 18
Sales/Inventory/Orders Standard Interface 18
How Price Uses the Sales Data 19
Data Fields 19

An Example 21
Technical Notes 23
Markdowns Taken Standard Interface 24
How Price Uses the Markdowns Taken Data 24
Data Fields 24
An Example 25
Technical Notes 25
Budget Standard Interface 26
How Price Uses the Budget Data 26
Data Fields 26
Technical Notes 26
Promotions Standard Interface 26
How Price Uses the Promotions Data 26
Data Fields 27
Technical Notes 27
Distribution Center Inventory Standard Interface 28
Distribution Center Allocation Standard Interface 28
Technical Notes 28
Merchandise Hierarchy Rename Standard Interface 29
Location Hierarchy Rename Standard Interface 29
Merchandise Hierarchy CDA Standard Interface 29
Location Hierarchy CDA Standard Interface 29
Items CDA Standard Interface Specification 29
Business Rule Instances Standard Interface 29
Technical Notes 30
Demand Parameters Standard Interface 30
How Price Uses the Parameters Data 30
Data Fields 30
Price Ladders Standard Interface 31
How Price Uses the Price Ladder Data 31
Data Fields 31
Technical Notes 32
Seasonalities Standard Interface 32
How Price Uses the Seasonalities Data 32
Data Fields 33

Price Interface Specifications 34
Merchandise Hierarchy Specification (ASH_MH_TBL) 34
Location Hierarchy Specification (ASH_LH_TBL) 37
Calendar Specification (ASH_CAL_TBL) 40
Items Specification (ASH_ITEMS_TBL) 41
Sales Specification (ASH_SALES_TBL) 42
Markdowns Taken Specification (ASH_MDTAKEN_TBL) 45
Budget Specification (ASH_BUDGET_TBL) 46
Promotions Specification (ASH_PROMO_TBL) 47
DC Inventory Specification (ASH_DCI_TBL) 49
DC Allocation Specification (ASH_DC_ALLOCATION_TBL) 50
MH Rename Specification (ASH_MHRENAME_TBL) 51
LH Rename Specification (ASH_LHRENAME_TBL) 52
MH CDA Specification (ASH_MH_CDA_TBL) 53
LH CDA Specification (ASH_LH_CDA_TBL) 55
Items CDA Specification (ASH_ITEMS_CDA_TBL) 57
Business Rule Instances Specification (ASH_BRM_INSTANCE_TBL) 59
Demand Parameters Specification (ASH_PARAMETER_VALUES_TBL) 60
Price Ladders Specification (ASH_PRICE_LADDERS_TBL) 61
Seasonalities Specification (ASH_SEASONALITY_MAPS_TBL and ASH_SEASONALITY_VALUES_TBL) 63

3. Standard Load

Introduction 66
Standard Load Process 67
Environment Customization File 68
Staging Script: pl_stage_file.sh 69
Load Script: pl_load_data.sh 70
Load Procedures 70
Load Merchandise Hierarchy 70
Load Location Hierarchy 71
Load Merchandise Table 71

Load Location Table 71
Load TClose Table 72
Load LTClose Table 72
Load MH Rename 73
Load LH Rename 73
Load Calendars 73
Load Items 74
Load Sales 74
Load Markdowns Taken 75
Load Budget 75
Load Promotions 76
Load Warehouses Table 76
Load DC Inventory 76
Load Warehouse Allocation 77
Load Business Rule Instances 77
Load Sendbacks 78
Load Model Start Date 78
Load Materialized Views 79
Load Scenarios 80
Load Internal Promos 80
Load Parameters 80
Load Price Ladders 81
Load Seasonalities 81
Load Collections Sendback 81
Load Collections Auto 82
Standard Load Error Handling 83
Error Handling Property File 84
Custom Errors 85
Error Handling Report 87
Standard Load Error Messages 88
Standard Load Dependency Tree 108
Standard Load Steps 111
Standard Interface Specifications for One-Time Data 112
Cross Products Information Standard Interface (ASH_CP_TBL) 112
Technical Notes 112

Cross Products Information Specification 112
Location Hierarchy Levels Standard Interface (ASH_LHL_TBL) .. 113
Technical Notes 113
LH Levels Specification 113
Merchandise Hierarchy Levels Standard Interface (ASH_MHL_TBL). 113
Technical Notes 114
MH Levels Specification 114
Cluster Levels Standard Interface (ASH_CSHL_TBL) 114
Technical Notes 114
CSH Levels Specification 115
Standard Dataset 116
Dataset Data 117
Modifying the Dataset 120
Sample Model Run Results 121

4. The Model Run

Introduction 124
The Optimization Engine 125
Optimization Engine Configuration 126
Settings for job.properties 126
Settings for delphi.properties 126
Optimization Run Prerequisites 129
Optimization Run Process 130
Optimization Run Scripts 130
bashjava.sh 130
checkRunSuccess.sh 130
closeCurrentJob.sh 131
enginectl.sh 131
generateErrorWorkQueue.sh 131
getCurrentJobID.sh 132
getCurrentJobStatus.sh 132
getEngineVersion.sh 132
initializeJob.sh 133
isDone.sh 133

jobHistory.sh ... 133
jobReport.sh ... 134
multiChunker.sh ... 134
plfrontendload.sh ... 134
plpostmodelrun.sh ... 135
plpremodelrun.sh ... 135
refreshForecastCache.sh ... 135
refreshSummaryCache.sh ... 136
runCalcEngine.sh ... 136
runChunker.sh ... 137
runInteractiveCE.sh ... 137
runMultiKPI.sh ... 137
runReport.sh ... 138
runStatsOnBatchOutput.sh ... 138
Load_statements.sql ... 138
FELOAD ... 138
PRERUN ... 139
POSTRUN ... 140
Summary Metrics ... 140
Monitoring an Optimization Run ... 142
Running Reports and Diagnosing Problems ... 144
Restarting a Run ... 146

5. Price Tools

PriceAdmin Commands ... 148
disableLogin ... 148
enableLogin ... 149
generateAlerts ... 149
generateSendback ... 150
isSummaryCacheLocked ... 150
listSendbackTypes ... 151
lockWorksheets ... 151
refreshForecastCache ... 152
refreshSummaryCache ... 152
registerAlerts ... 152

releaseSummaryCacheLock 153
unlockWorksheets 153

Contents

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 book assumes that you understand:

- configuration concepts
- retail metrics, concepts, and business processes

Price Documentation Set

For additional information about using Price, see the following:

- *ProfitLogic Price Installation Guide* for instructions on setting up the application server and database environment for Price as well as installing Price.
- *ProfitLogic Price Configuration Guide* for instructions on configuring the Price front end (user interface), loading data, and configuring and performing runs.
- *ProfitLogic Price User Guide* for information on how to use Price and on all its functionality and screens.
- *ProfitLogic Price Operations Guide* for instructions on weekly processes and maintaining Price.
- Price Online Help for instructions on accessing worksheets, key items, forecasts, reports, etc., and for using features such as What If and Optimize to Budget.
- If your installation of Price includes Merchant Desktop, see the Merchant Desktop Help for instructions on accessing Price, Price merchandise alerts, and enhanced reporting; using worksheets, What If, and Optimize to Budget features; and personalizing your desktop.

Technical Support

If you experience problems and need technical assistance, see your ProfitLogic Professional Services representative.

Conventions

Throughout this guide, the term *administrator* refers to the person(s) who configures and maintains the application and related features such as Merchant Desktop. In some cases the administrator will be someone at your company, and in other cases, the administrator will be someone from ProfitLogic Professional Services.

Convention	Description
`Monospace`	Represents text that you must type
Monospace italic	Indicates a variable for which you supply a value
Note:	Indicates exceptions and information that, if omitted, can cause you to perform one or more actions again
Caution:	Indicates danger to the integrity of the data or system

1 Introduction

The chapter contains the following:

- “About the Price Operations Guide” on page 2
- “What’s In This Book” on page 3

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 - Tools - describes the PriceAdmin tools.


ProfitLogic

2 Standard Interface

This chapter contains the following:

- “Merchandise Hierarchy Standard Interface” on page 9
- “Location Hierarchy Standard Interface” on page 12
- “Calendar Standard Interface” on page 14
- “Items Standard Interface” on page 16
- “Sales/Inventory/Orders Standard Interface” on page 18
- “Markdowns Taken Standard Interface” on page 24
- “Budget Standard Interface” on page 26
- “Promotions Standard Interface” on page 26
- “Distribution Center Inventory Standard Interface” on page 28
- “Distribution Center Allocation Standard Interface” on page 28
- “Merchandise Hierarchy Rename Standard Interface” on page 29
- “Location Hierarchy Rename Standard Interface” on page 29

- “Merchandise Hierarchy CDA Standard Interface” on page 29
- “Location Hierarchy CDA Standard Interface” on page 29
- “Items CDA Standard Interface Specification” on page 29
- “Business Rule Instances Standard Interface” on page 29
- “Demand Parameters Standard Interface” on page 30
- “Price Ladders Standard Interface” on page 31
- “Seasonalities Standard Interface” on page 32
- “Price Interface Specifications” on page 34

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.

Note: 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.”

Information about the Flexible Clustering Standard Interface can be found in the *Price Configuration Guide*.

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

Table 2-1 **Interface Specifications (Continued)**

Interface Specification	Required/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	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 11, 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 29.
- 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 11, 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 2-3 on page 13, 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 29.
- 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 13, the two records describing the hierarchy above Region FL2 are identical. Note that this consistency requirement applies to all three of the hierarchy fields (Key, ID, and Desc). Inconsistent values for hierarchy descriptions are a common reason why some location hierarchy records fail to load.
- Each node in a hierarchy can only have one parent node.
- The lowest level in the location hierarchy should be the level at which sales data is provided.
- The 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

Eight 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.

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 24.)
- 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

(see next page)

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. Retl. 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.
- 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.

Note: 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

Five 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
- OLD_TICKET_PRICE - the retail price prior to the markdown
- NEW_TICKET_PRICE - the new retail price after the markdown takes effect

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.

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.

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 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 ouput, 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.

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

Eleven fields describe each price ladder:

- PRICE_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_LADDER_TYPE - price ladders are expressed as either percent off (PO) 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. All other flags are displayed in the UI.

- LADDER_MARKDOWN_TYPE - Point-of-sale markdown (POS)or permanent markdown (PRM).
- LADDER_PRICE - If the price ladder type is PP, then this contains the price point values.
- LADDER_PCT_OFF - If the price ladder type is PO, then this contains the percentage off (a value between 0 and 1)

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.

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.

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 2-8 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

Table 2-8 Merchandise Hierarchy Standard Interface Specification (Continued)

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

Table 2-8 Merchandise Hierarchy Standard Interface Specification (Continued)

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

Location Hierarchy Specification (ASH_LH_TBL)

Table 2-9 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

Table 2-9 Location Hierarchy Standard Interface Specification (Continued)

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

Table 2-9 Location Hierarchy Standard Interface Specification (Continued)

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/N
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 2-10 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 2-11 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

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 2-12 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

Table 2-12 Sales/Inventory/Orders Standard Interface Specification (Continued)[1]

Attribute Name	Attribute Description	Data Type	Maximum Length	Nullable Y/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
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 2-13 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
OLD_TICKET_PRICE	Previous retail price	Decimal	22,2	Y
NEW_TICKET_PRICE	New retail 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.

Budget Specification (ASH_BUDGET_TBL)

Table 2-14 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 2-15 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

Table 2-15 Promotions Standard Interface Specification[1] (Continued)

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
PROMO_TYPE	Type of the promotion (floor, ceiling, or unrestricted). This column is mandatory.	String	50	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 2-16 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 2-17 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
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 2-18 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 2-19 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 2-20 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

Table 2-20 Merchandise Hierarchy CDA Standard Interface Specification[1] (Continued)

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
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 2-21 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

Table 2-21 Location Hierarchy CDA Standard Interface Specification[1] (Continued)

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
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 2-22 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

Table 2-22 Items CDA Standard Interface Specification (Continued)[1]

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
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 2-23 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	Y
MERCHANDISE_LEVEL	ID for this level of the hierarchy	String	50	Y
LOCATION_KEY	Key for this level of the hierarchy	String	50	Y
LOCATION_LEVEL	ID for this level of the hierarchy	String	50	Y
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	Y
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	N

Demand Parameters Specification (ASH_PARAMETER_VALUES_TBL)

Table 2-24 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

Table 2-24 Demand Parameters Standard Interface Specification (Continued)

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/N
AS_VERSION_NUMBER	The version number for the current run of the ouput, which is set by APC and can be used to track versions.	String	20	Y

Price Ladders Specification (ASH_PRICE_LADDERS_TBL)

Table 2-25 Price Ladders Standard Interface Specification [1]

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
PRICE_LADDER_ID	Identifies the price ladder.	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_LADDER_TYPE	Percentage Off (PP) or Price Point (PP)	String	3	Y
PRICE_LADDER_DESC	Price ladder name displayed in the UI.	String	50	Y

Table 2-25 Price Ladders Standard Interface Specification (Continued)[1]

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MODEL_FLAG	R = price ladder used for optimization. All other flags are displayed in the UI.	String	2	Y
LADDER_MARKDOWN_TYPE	Point-of-sale markdown (POS) or permanent markdown (PRM)	String	3	Y
LADDER_PRICE	If price ladder type is PP, the price point values. (If this option used, then LADDER_PCT_OFF is null.)	Decimal	7,2	Y
LADDER_PCT_OFF	If price ladder type is PO, the percentage off (a value between 0 and 1). (If this option is used, then LADDER_PRICE is null.)	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 2-26 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 2-27 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.

3 Standard Load

This chapter contains the following:

- “Standard Load Process” on page 67
- “Standard Load Error Handling” on page 83
- “Standard Load Dependency Tree” on page 108
- “Standard Interface Specifications for One-Time Data” on page 112
- “Standard Dataset” on page 116

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.

Note: 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
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

Table 3-1 Flat Files Names

Example File Name	File Content
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

The two scripts are located in %DATABASEROOT%/%VENDOR%/schema/scripts, where the database root directory is set at the time of installation and the vendor is either DB2 or Oracle. 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. (Use this script if you need to customize the load dependency tree. If you do not need to optimize the load procedure, you can use a flat dependency tree by executing pl_load_client.sh.)

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

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:

-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:

-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 procedures for Flexible Clustering can be found in the *Price Configuration Guide*.

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

Note: 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

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 excepts 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.

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.

Note: 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 108).

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

Table 3-2 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 3-3 Table Columns for "BAD" Table

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 Property 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 page 84.

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 3-5, “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

For DB2:

```
db.type=db2
db.driver=com.ibm.db2.jcc.DB2Driver
db.url=jdbc:db2://<db_host>:<db_port>/<db_name>
db.username=<db_username>
db.password=<db_password>
```

where

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 111.

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:

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 3-4 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

Table 3-4 Sample Standard Load Data Validation Report (Continued)

LOAD_PROCEDURE	SOURCE	DATA_ VALIDATION_ STATUS	NUM_ BAD_ RECORDS
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
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. If you are running DB2, you will not see these error messages on your system console. Instead, you should refer to

<DB2InstanceHome>/sqllib/db2dump/db2diag.log to see the error message code and text. Any other exceptions that occur during the run will also be contained in this file.

Table 3-5 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.
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	Mandatory columns in the cross-products information table (ASH_CP_TBL) contain null values.
202	A duplicate INTERSECT_NAME has been found in the cross-products information table (ASH_CP_TBL).

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
203	Invalid attribute values have been found in the cross-products information table (ASH_CP_TBL).
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.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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	
1100	A mark-up cannot be loaded into the markdowns taken table (ASH_MDTAKEN_TBL).
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 old ticket price has a value of NULL. An old ticket price cannot have a value of NULL.
1106	In the markdowns taken table (ASH_MDTAKEN_TBL), at least one new ticket price has a value of NULL. A new ticket price cannot have a value of NULL.
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.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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.
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.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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.
1302	A record in the items table (ASH_ITEMS_TBL) contains a location that is not found in the location hierarchy.
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.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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.
Load DC Inventory Errors	
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.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
1604	The new location key in the location hierarchy rename table (ASH_LHRENAME_TBL) contains duplicate values.
1605	The new location key in the location hierarchy rename table (ASH_LHRENAME_TBL) is already present in the location hierarchy.
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) contains an entry for LEVEL_SQC with a value < 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.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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 vlaue 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.
2605	The new merchandise key in the merchandise hierarchy rename table (ASH_MHRENAME_TBL) is already present in the merchandise hierarchy.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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.
3541	MERCHANDISE_HIERARCHY_TBL is empty.
3551	LOCATION_HIERARCHY_TBL is empty.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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.
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.
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.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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 wek.
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 3-5 Standard Load Error Messages (Continued)

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.
Load BRM Rules Errors	
3801	The BRM_RULE_DEFINITION_TBL is empty and needs to be populated.
Load Flexible Clustering Errors	
3901	Cluster Definition Load has invalid Location LOCATION_CLIENT_ID. It should match what is defined in LOCATION_HIERARCHY_TBL <client_load_id>.
3902	The Cluster Mapping Load has invalid merchandise MERCHANDISE_CLIENT_ID. It should match what is defined in MERCHANDISE_HIERARCHY_TBL <client_load_id>.
3903	The Cluster Mapping Load has invalid cluster set CLUSTER_SETCLIENT_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 <Cluster Set Client_id>.
3905	Cluster to Location mapping warning: The following location is not contained in every cluster <location client id>.
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.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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.
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.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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 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.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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.
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 Errors	
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.
5100	Invalid input number for number of ITEM_DATA partitions.
5101	The ITEM_DATA table is empty.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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.
5108	The ITEM_DATA tabel 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	
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.

Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
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.

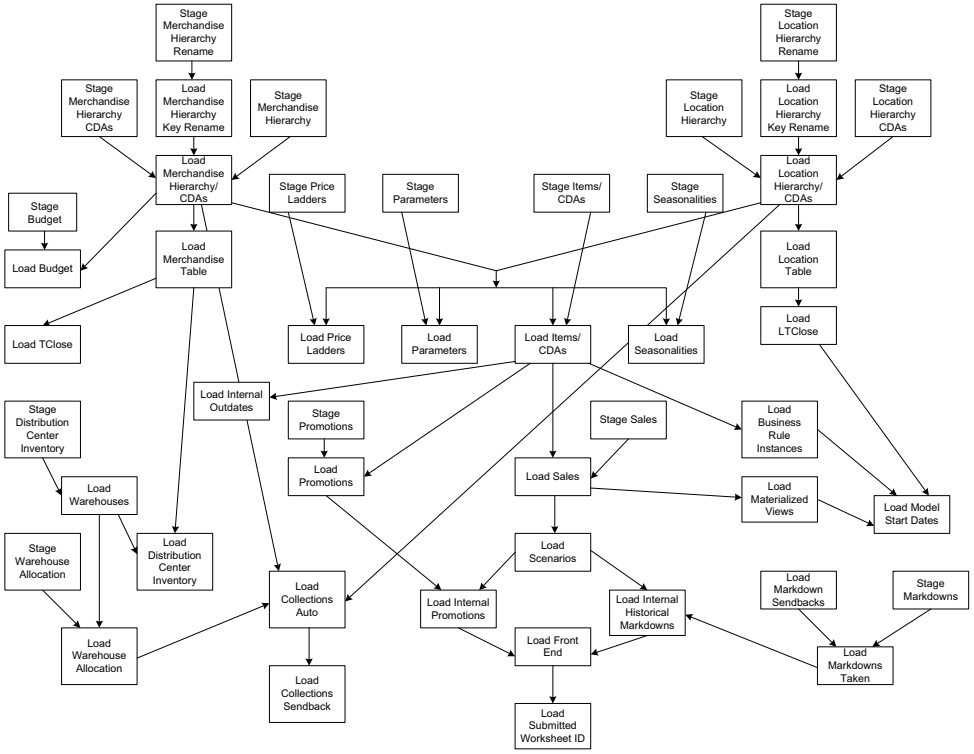
Table 3-5 Standard Load Error Messages (Continued)

Number	Error Message
8208	The found price ladder type is not in PT, PO, or PP.
8209	The model flag cannot be NULL.
8210	The laddr MD type cannot be NULL.
Load Seasonalities Error Messages	
8301	The merchandise key in ASH_SEASONALITY_MAPS_TBL cannot be null.
8302	The merchandise level in ASH_SEASONALITY_MAPS_TBL cannot be null.
8303	The location key in ASH_SEASONALITY_MAPS_TBL cannot be null.
8304	The location level in ASH_SEASONALITY_MAPS_TBL cannot be null.
8305	Merchandise found in ASH_SEASONALITY_MAPS_TBL that does not exist in MERCHANDISE_HIERARCHY_TBL.
8306	Location found in ASH_SEASONALITY_MAPS_TBL that does not exist in LOCATION_HIERARCHY_TBL.
8307	A NULL priority 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.

Standard Load Dependency Tree

The load script loads data in the order specified in the following dependency tree. Figure 3-1, Standard Load Dependency Tree (General), shows the dependency tree that is generally used. The dependency tree used for Flexible Clustering can be found in the *Price Configuration Guide*.

Figure 3-1 Standard Load Dependency Tree (General)

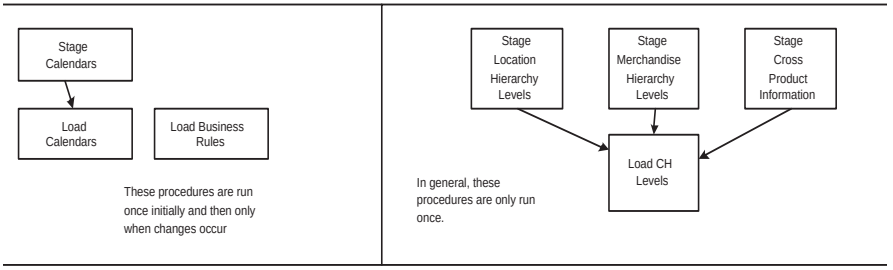


In addition, there are load procedures that are only run once

- 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 3-6 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

Table 3-6 Dependency Tree Procedures (Continued)

Dependency Tree Procedure	Procedure Name
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 3-6 Dependency Tree Procedures (Continued)

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

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 and is either OPTIMIZATION, SALES, WORKSHEET, or CLUSTER. 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.

Cross Products Information Specification

Table 3-7 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, or CLUSTER)	String	50	N
MERCHANDISE_LEVEL	The defining level within the hierarchy	String	50	N

Table 3-7 Cross Products Information Standard Interface Specification

Field Name	Field Description	Data Type	Maximum Length	Nullable Y/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 3-8 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 3-9 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 3-10 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 2,500 items and 200 collections. 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 3-11 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
Location Hierarchy Levels (ASH_LHL_TBL)	Five levels: Chain Company Zone Price zone Store

Table 3-11 Price Dataset Data (Continued)

Data Interface Table	Data Characteristics
Cross Products Information (ASH_CP_TBL)	MH-LH intersections: *Optimization* Product key (merchandise level) Prize zone (location level) *Worksheet* Department (merchandise level) Chain (location level) *Sales* Product key (merchandise level) Store (location level)
Merchandise Hierarchy (ASH_MH_TBL)	34,471 merchandise items at the product key level
Location Hierarchy (ASH_LH_TBL)	6 price zones 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)	120 price ladders

Table 3-11 Price Dataset Data (Continued)

Data Interface Table	Data Characteristics
Seasonalities (ASH_SEASONALITY_MAPS_TBL) and (ASH_SEASONALITY_VALUES_TBL)	300 seasonality settings
Collections (No interface or staging table)	200 collections
Business Rule Instances (ASH_BRM_INSTANCE_TBL)	3,015 instances of default rules: 500 OUT_DT 1,000 INVENTORY_TARGET 1,500 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     94975
Green       2126

Message Breakdown
```

Category	Resource	Item Count	Message
Item Data	engine.status.didyma.noStartDate	24848	This item has no Start Date
Item Data	engine.status.didyma.noTicketPrice	24848	This item has no Ticket Price
Item Data	engine.status.didyma.noFullPrice	24552	This item has no Full Price
Item Data	engine.status.agorai.badLastStoreCount	38	Last week of historical activity does not have a good Store Count
Inactivity	engine.status.agorai.noUsefulSales	476	Data too dirty to determine demand: zero useful sales found
Error	engine.status.didyma.itemBuild	12424	Error building optimization request for item
Error	engine.status.didyma.collectionBuild	7789	Error building Optimization Request for collection
Not Recommended	engine.status.agorai.sellsOutWithoutChanges	33	Sells to target without changes; Markdown not recommended
Not Recommended	engine.status.agorai.notRecommended	5	Markdown not recommended for this week

Warning	engine.status.agorai.priceAboveFull	14	A relative price is higher than the full price: 1.082
Informational	engine.status.agorai.version	2074	Agorai version: (private build)

4 The Model Run

This chapter contains the following:

- “Introduction” on page 124
- “The Optimization Engine” on page 125
- “Optimization Run Prerequisites” on page 129
- “Optimization Run Process” on page 130
- “Monitoring an Optimization Run” on page 142
- “Running Reports and Diagnosing Problems” on page 144
- “Restarting a Run” on page 146

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, writing results back to the database, and calculating 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.size - defines the approximate minimum number of items per chunk. The optimal value is related to the anticipated memory use per thread on the Optimization Engine worker machines, which may vary by dataset. The larger the memory usage, the fewer workers that can run on a single machine. If the application host is memory-limited, try reducing the chunk size.

  Default value = 10,000.
- 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 10 minutes.

  Default value = 10
- 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.

Table 4-1 delphi.properties Settings (Continued)

Property	Description
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=	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.[1]
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.
strategy.modelvalues=list	Used for performance tuning for IR_MODEL_VALUES.
strategy.pastticketprices=list	Used for performance tuning for IR_PAST_TICKET_PRICES.

Table 4-1 delphi.properties Settings (Continued)

Property	Description
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.

1. See “Inference Rule Access” on page 49 for more information on all strategy properties.

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.

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.)

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.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.

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.

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.

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
- n - the number of threads being used
- c - processsing 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.

runCalcEngine.sh

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

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] <full_path_to_configroot>

where

- n - number of parallel processes

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

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_suummaries, forecast_activities, rtm_history, rtm_status, and rtm_status_arguments.

Load_statements.sql

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

FELOAD

During the FELOAD portion of the optimization run, the ITEM_DATA table is updated with new data.

FELOAD consists of the following steps:

1. The SCENARIOS_TBL table is loaded.
2. The INTERNAL_PROMO table is loaded.
3. The INTERNAL_HIST_MKDNS_TBL table is loaded.
4. The BRM attributes are cached into the CDW_BRM_ATTRIBUTES_VW table.
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.

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 buisness rules exploded to th eitem 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.

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 WIF_SCENARIO_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. BRM_ATTRIBUTE_VALUE_TBL is populated.
10. HIST_MARKDOWNS_MODELRUN_TBL (HIST_MARKDOWNS cache) is populated.
11. All table statistics are updated.
12. 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. Statistics are collected on the P4P_FORECAST_DATA table.
4. All dropped indexes are restored on the P4P_FORECAST_DATA table by the RestoreTable procedure.
5. Statistics are collected on the P4P_FORECAST_DATA table.
6. The TMP_P4P_FORECAST_DATA table is truncated in preparation for the load.
7. The ITEM_DATA table is updated, based on the results of the model run, with COLLECTION_NAME and MARKDOWN_STATUS_KEY.
8. Statistics are collected on the ITEM_DATA table.
9. Data from the ITEM_DATA table is used to update RECOMMENDED_COLLECTION_FLAG and WORKSHEET_ID in P4P_COLLECTION.
10. Statistics are collected on the P4P_COLLECTION table.
11. All invalid views are recompiled.

Summary Metrics

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

The P4P_WORKSEET_SUMMARIES table stores the aggregate data for all summary metrics for all worksheets. The P4P_WORKSEET_SUMMARIES table is indexed after 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 now 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** command is invoked

The cached table is rebuilt when:

- the application server is running and the configuration *has* changed and the **refreshSummaryCache** 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, 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 GUI 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 now 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 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. If a worker process exits with an error, it is good practice to have another worker process start up automatically in its place.

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 126 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.

5 Price Tools

This chapter contains the following:

- PriceAdmin Commands
  - — “disableLogin” on page 148
  - — “enableLogin” on page 149
  - — “generateAlerts” on page 149
  - — “generateSendback” on page 150
  - — “isSummaryCacheLocked” on page 150
  - — “listSendbackTypes” on page 151
  - — “lockWorksheets” on page 151
  - — “refreshForecastCache” on page 152
  - — “refreshSummaryCache” on page 152
  - — “registerAlerts” on page 152
  - — “releaseSummaryCacheLock” on page 153
  - — “unlockWorksheets” on page 153

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 Production Run 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 weblogic.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.price.PriceAdmin**. For example:

A WebLogic run-based command using port 7033

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

A WebLogic p4gui-based command

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

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

disableLogin

Description: The **disableLogin** 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:

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

refreshSummaryCache --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, --continue on failure	processing continues on failure

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

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.