Oracle® Retail Markdown Optimization

Configuration Guide

Release 13.4

December 2012

ORACLE®

Oracle® Retail Markdown Optimization Configuration Guide, Release 13.4

Primary Author: Judith Meskill

reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

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

Contents

3 Business Rule Property Manager

4 Inference Rules

5 What If

6 Configurable Data Attributes

7 Internationalization

8 Pricing Group Management

9 Flexible Store Clustering

10 Understanding the Application (GUI) Configuration

11 Configuring the Application (GUI)

12 Configuration Properties Files

13 Standard Reports

14 RDM Data Mapping

15 Metadata Metrics

A Standard Columns

Send Us Your Comments

Oracle Retail Markdown Optimization Configuration Guide, Release 13.4

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at http://www.oracle.com.

Preface

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

Audience

This document is intended system administrators who configure and manage Markdown Optimization.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following documents in the Markdown Optimization Release 13.4 documentation set:

- *Oracle Retail Markdown Optimization Administration Guide*
- *Oracle Retail Markdown Optimization Data Model*
- *Oracle Retail Markdown Optimization Grid Designer User Guide*
- *Oracle Retail Markdown Optimization Implementation Guide*
- *Oracle Retail Markdown Optimization Installation Guide*
- *Oracle Retail Markdown Optimization Operations Guide*
- *Oracle Retail Markdown Optimization Release Notes*
- *Oracle Retail Markdown Optimization User Guide*
- *Oracle Retail Merchandise Planning and Optimization Licensing Information*

or in the Oracle Retail Analytic Parameter Calculator Markdown Optimization documentation set:

- *Oracle Retail Analytic Parameter Calculator Markdown Optimization Configuration Guide*
- *Oracle Retail Analytic Parameter Calculator Markdown Optimization Installation Guide*
- *Oracle Retail Analytic Parameter Calculator Markdown Optimization Release Notes*
- *Oracle Retail Analytic Parameter Calculator Markdown Optimization User Guide*

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

`https://support.oracle.com`

When contacting Customer Support, please provide the following:

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

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 13.4) or a later patch release (for example, 13.4.1). If you are installing the base release, additional patch, and bundled hot fix releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch and bundled hot fix releases can contain critical information related to the base release, as well as information about code changes since the base release.

Oracle Retail Documentation on the Oracle Technology Network

Documentation is packaged with each Oracle Retail product release. Oracle Retail product documentation is also available on the following Web site:

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

(Data Model documents are not available through Oracle Technology Network. These documents are packaged with released code, or you can obtain them through My Oracle Support.)

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

Conventions

The following text conventions are used in this document:

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

1

Getting Started

This chapter provides an orientation to MDO and contains the following sections:

Introduction

The Markdown Optimization application is deployed in a distributed, replicated architecture with a single system image and a single point of control and configuration. The deployment of changes to the configuration is based on the fielding of a comprehensive set of configuration changes. Change management supports the rolling back of changes if undesirable side effects occur.

The core functionality of MDO is the weekly model run.The goal of the MDO configuration is the creation of a model run that completes successfully in a timely fashion and produces valid markdowns and forecast recommendations in fulfillment of the retailer's requirements.

The *Oracle Retail Markdown Optimization Configuration Guide* is a reference that contains the technical details about the configuration of the application in support of this functionality.

Note: This Configuration Guide is designed to be used in conjunction with the entire MDO documentation set. In particular, you should consult the following four reference guides for detailed information about various aspects of the configuration.

- *Oracle Retail Markdown Optimization Implementation Guide*
- *Oracle Retail Markdown Optimization Operations Guide*
- *Oracle Retail Markdown Optimization Administration Guide*
- *Oracle Retail Analytical Parameter Calculator for Markdown Optimization User Guide*

MDO Environment

A number of environments are involved in Markdown Optimization deployments:

- **Development environment**. Used internally by an implementer to try out initial configurations or enhancements.
- **Test environment.** Used internally to verify and validate correct functionality prior to deployment to a staging environment or a production environment.
- **Staging environment.** Used for testing, validating, and verifying changes before they are applied to a production environment.
- **Production environment.** The environment that provides the application to end-users and in which the weekly batch process is run. This environment is change-controlled and monitored.

A typical Markdown Optimization environment consists of the following integrated pieces:

- Oracle database
- Application server: WebLogic
- Markdown Optimization application, consisting of software, database schema, baseline configuration, and application server setup
- Environment-specific configuration:
  - Front end configuration
  - Database schema updates
  - Custom integration points
- IT Infrastructure
  - Hardware and systems software
  - Networking setup
  - Database instance creation
  - Database clients
  - Application server base installation
  - User accounts
  - User access

Getting Started with Configuring Markdown Optimization

Note: The configuration process itself is covered in detail in the *Oracle Retail Markdown Optimization Implementation Guide.* This Guide provides the technical details in support of the implementation.

Once you have installed Markdown Optimization, you are ready to configure it to perform model runs in a production environment. Optimization runs produce forecasting and markdown recommendations.

The best approach to configuring Markdown Optimization is to get a basic production environment up and running. Once that is done, you can customize the application

incrementally as needed to meet specific retailer requirements and as business requirements change.

Configuration Process

The basic process, at a high level, consists of the following steps:

1. Plan and prepare the product environment.
2. Collect information about business requirements, including the level of sales data and optimization, the level of worksheets, the contents of the weekly data feed, the content of the weekly sendback files, business rule configuration, markdown validation configuration, model start date configuration, price ladder configuration, promotions configuration, markdown calendar definition, and the GUI configuration (worksheets, columns, and metrics).
3. Install the application. See the *Oracle Retail Markdown Optimization Installation Guide*.
4. Load historic data from the retailer.
5. Analyze the business data and develop parameters for the forecasting model (Analytical Services).
6. Load weekly production data from retailer.
7. Do the first model run, using primarily default application settings.
8. Use the diagnostic tools to assess and troubleshoot the model run and modify the configuration as needed.
9. Perform weekly model runs in a production environment.
10. Configure the application incrementally according to retailer requirements.
11. Do additional model runs to test the incremental changes to the configuration.
12. Configure front end metrics.
13. Build front end grids.
14. Configure standard reports.
15. Create user roles and actions.

Markdown Optimization Required Skill Sets

The following skill sets are required to run Markdown Optimization.

Operations

The Operations staff is responsible for the daily operation of the application. Skills include:

- Basic UNIX knowledge. Markdown Optimization provides a UNIX command line interface to all jobs that require scheduling.
- An understanding of relational database management systems and the use of SQL.
- An understanding of enterprise schedulers. The weekly batch process consists of large number of jobs with complex dependencies that are best managed using a scheduler.

Systems Administration

The systems administration staff is responsible for daily UNIX administration. Skills include:

- Familiarity with volume management.
- Setting up enterprise backups for file systems and databases. Performing nightly hot backups. An understanding of relational database software. Tape backups and restores.
- Tape rotation.
- Operating system installation.
- Patch application.
- Security-hardened OS configurations.
- Kernel parameter tuning for DBMS and J2EE servers.
- Installation and configuration of ssh, rsync, bash, Gnu utilities, Python, and sudo.
- SAN storage configuration and management.
- Shell scripting.

Database Administration

The database administration staff is responsible for critical daily RDBMS management and troubleshooting. Skills include:

- Management of large databases (50 GB - 500GB).
- Database backup and recovery, including monitoring, backups, and restores.
- Data migration between test and production environments.
- Database tuning in response to problems or to enhance performance.
- Database layout, including creating tablespaces and partitioning.
- Storage management, including data volumes, log volumes, and index volumes.

Network Administration

The network administration staff is responsible for managing network equipment. Skills include:

- TCP/IP.
- Hardware load balancing for high availability and horizontal scalability.
- If SSL is being used to encrypt data, SSL Hardware Acceleration.
- Configuration of services on Cisco content switches or F5 BigIP switches.
- Managing routers with traditional ACLs.
- Managing firewalls for granular ACLs.
- Managing VPN tunnels.
- Understanding interoperability issues.

Storage Administration

The storage administration staff is responsible for managing a SAN environment and should be familiar with basic LUN management.

Application Server Administration

The application server administration staff should be familiar with the management of WebLogic. Skills include:

- Deploying and managing Java applications in a J2EE environment.
- Installing and configuring application server software.
- Configuring and tuning JDBC drivers and connection pools.
- Configuring JMS server and JMS queues.
- Administering multiple servers and clusters from a J2EE console.
- Deploying Web applications and EJB applications in the J2EE server.
- Configuring and tuning threads and message-driven beans.
- Monitoring and troubleshooting J2EE servers.
- Database development experience with Oracle.
- XML knowledge.
- Familiarity with shell scripts, Perl, and UNIX/AIX environments.

Configuration Points

The following are the specific functional areas of MDO that can require configuration:

- The extraction, transformation, and loading of retailer data.
- The demand parameters, such as seasonality and price effects, that are used to determine optimal markdown recommendations.
- The roles and permissions assigned to users.
- The business rules that determine which data is used by the application for an optimization. Business rules specify retailer constraints that are used by MDO to determine markdowns and forecasting.
- The inference rules that define queries specifying particular views into the database that provide customization points for MDO. An inference rule corresponds to a specific business policy.
- The user interface functionality and display.
- The retailer's metrics.
- The model run customization: ir.sql and load_statements.sql.
- The performance of Calculation Engine, which computes optimizations and forecasts for MDO.
- The parameters in config.properties and config.properties.

Change Management

The following table details the types of changes that can occur to the configuration after it has been implemented.

Table 1–1 Typical Markdown Optimization Configuration Changes

Change	Example
Business rules that are managed by end users or administrators	Changing the outdate for an item.
Data feeds that are not part of the standard weekly load	Changes or additions to price ladders.
Patches or hot fixes	Correcting a defect or adding a new feature.
Changes to scheduled processes resulting from a retailer business need	Change to arrival time for weekly data feed. Change to database backup schedule. Change to sendback schedule.
Changes to hardware or software infrastructure	OS security patch. Hardware replacement or addition. Application server patch. DBMS software patch.
Changes or enhancements to the application configuration	New custom metric added as column to a worksheet. New business rule value.
Changes to data hierarchies that impact the functional or analytical configuration	Major merchandise reclassification. Major changes to climate zones.
Updates to analytical parameters	Changes to seasonality parameters to reflect recent history of sales data.

Implementing and Deploying Changes

The following process is considered the best practice when changing the configuration.

1. Refresh a segregated development environment with a recent copy of the production environment. This should include:
   - Restoring the database from a backup, export, or disk image.
   - Installing all configuration files as they are in the production environment.
   - Verifying the correct functioning of the application in the production environment. A performance baseline should be captured if changes are being made that could impact the performance of the standard load or the model run.
2. Ensure that the configuration prior to the change is saved to source control.
3. Make changes in the production environment and unit test to verify that they are functioning correctly.
4. To test the configuration changes adequately, execute a full weekly cycle, including loading data, running the model, taking markdowns in the application, and running sendbacks. Multiple weekly cycles may be appropriate.
5. After changes have been verified in the development environment, they should be checked into source control.
6. Create a staging environment with the most recent copy of the production environment and capture performance baseline data.

7. Apply the changes checked into source control to the test environment, using the same method that will be used to apply the changes to the production environment.
8. Perform integration testing of the changes. A general smoke test of the application function and model run should be performed. The performance should be evaluated relative to the baseline data.
9. Most production changes should be applied in the interval between the final weekly sendback and the upcoming data load and model run. A full backup of the production database and application environment should be taken prior to applying any changes to production.

It can occasionally be necessary to revert a configuration if the implemented change does not work as expected. To do this, restore the backup image.

Applying Patches

Oracle makes changes to the application available as a configuration build. This build should be stage prior to being applied to a production environment. The build file should be copied to the INSTALL_BASE directory of the environment to be updated.

Use the following command to apply the new build:

INSTALL_BASE/integration/tools/field.sh <path_to_build_file>/<build_name>.tgz

This command executes the following steps:

- Backup
- Cleans up old files
- Unpacks the tar archive
- Specialization
- Expands templates
- Automatic editing changes
- Stops servers
- Cleanup and synch
- Updates database
- Applies one-time and regular schema updates
- Updates the BRPM configuration
- Restarts the servers
- Updates user roles
- Automation set-up
- Cron set-up

2

User Management

The User Management utility is accessed from the MDO Administration menu and is used to configure user access to the application.

This chapter contains the following sections:

Introduction

User Management (UM) is a utility that lets you create, modify, and remove user accounts from a central location. The User Management utility is installed automatically when you install the application.

As part of the installation, the root user is created with only the Price Admin role. The root user's credentials must be added to the Oracle Wallet before the installation and provided during the installation. The root user is responsible for creating the other users and assigning roles to these users.

In addition, the griduser1 user is created as part of the installation.

Each user who accesses the application must have a user account. Each user account is assigned one or more roles that determine the types of functions the user can perform with the application.

Single sign-on is supported so that users can access from the Main Menu the MDO GUI, BRPM, UM, and Seasonality Manager without additional authentication.

About User Roles and User Actions

Roles are defined by a specific set of user actions. The actions that define each role serve to delimit the activities a user can perform. All actions are self-contained. For example, Write does not imply Read. So a role must include all the actions that are necessary for complete functionality. If a role is assigned at a specific level in the hierarchy and that hierarchy level is removed, then the role is removed.

Markdown Optimization comes with a default set of roles, loaded into ROLE_ACTION_TBL. Default action are assigned to the roles. These cannot be deleted. For more information on Business Rule Property Manager roles and actions and Seasonality Manager roles and actions, see those respective chapters.

Note: You must first load the Merchandise Hierarchy and Location Hierarchy before you assign roles.

The following table lists the MDO roles and the default actions assigned to those roles.

Table 2–1 MDO Roles and Default Actions

Role	Default Action
PRICE_APPROVER	PRICE_APPROVE
PRICE_SUBMITTER	PRICE_SUBMIT
	PRICE_COMMENTS_EDIT
PRICE_USER	PRICE_MARKDOWNS_VIEW
	PRICE_MAINTAINING_MERCHANDISE_VIEW
	PRICE_BRM_VIEW
	PRICE_USER_PROFILE_VIEW
	PRICE_REPORTS_VIEW
	PRICE_GUARD
	PRICE_ITEM_INFO_VIEW
PRICE_VIEWER	PRICE_VIEW
BRM_PRICE_EDIT	BRM_PRICE_EDIT
	PRICE_SEASONALITY_EDIT
BRM_PRICE_VIEW	BRM_PRICE_VIEW
	PRICE_SEASONALITY_VIEW
BRM_PROFITLOGIC_EDIT	BRM_PROFITLOGIC_EDIT
BRM_PROFITLOGIC_VIEW	BRM_PROFITLOGIC_VIEW
WHAT_IF_SERVICE_USER	MDO_WHAT_IF_SERVICE_EXEC

The MDO roles are defined as follows.

- PRICE_ADMIN_EXEC – can run the PriceAdmin commands.
- PRICE_APPROVER – has read-only access to worksheets and can approve submitted worksheets at the specified level in the hierarchy, but cannot submit worksheets.
- PRICE_SUBMITTER – can submit worksheets at the specified level in the hierarchy.
- PRICE_USER – allows access to the UI.
- PRICE_VIEWER – has read-only access to worksheets at the specified level in the hierarchy.
- BRM_PRICE_EDIT – can edit a business rule through the UI at the item level or higher.

- BRM_PRICE_VIEW – can view a business rule through the UI at the item level or higher.
- BRM_PROFITLOGIC_EDIT – can edit administrative business rules.
- BRM_PROFITLOGIC_VIEW – can view administrative business rules.
- WHAT_IF_SERVICE_USER – similar to the PRICE_USER role, but for the web service.

Significant MDO actions are defined as follows.

- PRICE_COMMENTS_EDIT – can edit tool tips.
- PRICE_GUARD – guards against illegal access to the application.
- MDO_WHAT_IF_SERVICE_EXEC – provides access to the web service.
- PRICE_MAINTAINING_MERCHANDISE_VIEW_NO_PG_EDIT – provides read-only access to the Pricing Group Manager (only the action is available by default).
- PRICE_ITEM_INFO_VIEW – can view item information.
- PRICE_SEASONALITY_VIEW – can view seasonality curves but cannot override or change mappings.
- PRICE_SEASONALITY_EDIT – can edit seasonality curves. This permission only applies to hierarchies that the user has permissions to edit.

Roles are assigned to users with restrictions that are defined at or above a specific node of the merchandise hierarchy and the location hierarchy. The scope of actions can be across the merchandise and location hierarchies. The scope must be defined at or above the class planning level.

The sample file, "Role Assignment Sample XML File" provides an illustration of defining the scope.

About User Management Roles

User accounts with user management roles have access to features such as creating users, assigning roles, removing user accounts, resetting passwords.

When a user with a user management role logs on, a link to the User Management utility appears on the Main Menu.

The following list describes the default User Management roles:

- UM_READ_ONLY_ADMIN – This role allows read-only access to the User Management utility. This role has privileges to view the list of users and their roles and hierarchy levels, but not to create new user accounts or modify or inactivate existing ones.
- UM_ROLE_ASSIGN_ADMIN – This role allows assigning new roles (and related hierarchy levels) to existing user accounts, but it does not allow the creation of new user accounts.
- UM_USER_ADMIN – This role allows creating new user accounts, but it does not allow the assignment of roles to the new accounts.

User Management Bulk Loader Utility

If you are creating a small number of user accounts using the default roles, you can create those accounts using the application UI. (For more information on using the

User Management utility, consult the Markdown Optimization Online Help.) However, if you want to create user accounts for a group of users all at one time, you can use the User Management bulk loader utility.

Prior to running the User Management bulk loader utility, you must:

- Set the jndi.properties. The jndi.properties file, which is located in <installed>/modules/tools/conf/jndi.properties, specifies the initial context factory and the url where the JNDI lookups are carried out.

  For WebLogic, typical values are:

  app.server.home=C:/<WLS_HOME>/wlserver_ 10.3/serverjava.naming.factory.initial=weblogic.jndi.WLInitialContextFactoryjava. naming.provider.url=t3://localhost:7001

- Make sure that usermanagement.ear, suiteproperties.ear, and common4p.ear are deployed on the running application server.

For further information, consult the WebLogic documentation.

Users and Roles

You need to create and validate (using a tool like XML Spy) three XML files containing entries for Users, Roles, and Role Assignments.

- The user file contains user names. All user names must be unique. The schema includes a flag that indicates whether or not the password should be hashed.
- The Roles file contains the possible roles that can be assigned. All role keys must be unique. The action key attributes must be loaded into the database before the bulk loader utility can be used. All elements and attributes must be lower case.
- The Role Assignment file contains user names and the role or roles associated with the user name. The user names must be loaded into the database before this file can be processed by the bulk loader utility. All elements and attributes must be lower case. The merchandise ID and the Location ID are provided by a pipe-delimited string of CLIENT_LOAD_ID, as found in the MERCHANDISE_ HIERARCHY_TBL or LOCATION_HIERARCHY_TBL. For example, to assign a user to a certain department of merchandise:

  CHAIN COMPANY DIVISION DEPARTMENT merchandise attribute in .xml

  ---

  0 1 123 8765 1 | 123 | 8765

  0 1 22 789 1 | 22 | 789

The information in the three files is loaded into database tables by the bulk loader. (Users and Role Assignments can be added or modified using the application UI. Roles can only be added or modified using the bulkloader.)

The XML Files

The XML schemas and samples of the three required XML files can be found in <installed>/config/Price/security.

Table 2–2 User Management XML Files

Schema	Sample	Database Table
user-set.xsd	price_user_set.xml	USERS_TBL

Table 2–2 (Cont.) User Management XML Files

Schema	Sample	Database Table
role-set.xsd	price_role_set.xml	ROLES_TBL
role-assignment-set.xsd	price_assignment_ pricedataset.xml	USER_RESOURCE_ROLE_TBL

Standard Load Prerequisites

Before you run the bulk loader, you must have run the standard load so that the merchandise hierarchy table (ASH_MH_TBL) and the location hierarchy table (ASH_LH_TBL) have been populated. (For more information on the standard load, see the *Oracle Retail Markdown Optimization Operations Guide*).

Shell Script

The shell script for running the User Management bulk loader utility is located in <installed>/modules/tools/bin/bulkloader.sh.

Usage:

-apphome <directory>	application server home directory
-assignfile <filename>	file for loading role assignments
-rolefile <filename>	file for loading roles
-userfile <filename>	file for loading users
-verbose	print debug information
-useralias <admin alias>	administrative user's alias, stored in the CSM wallet

The bulkloader.sh script requires the CSM_HOME environment variable (which is set during installation).

To run the shell script (an example):

Note that the three files can be loaded separately or at the same time.

$bash **bulkloader.sh** -apphome /usr/local/bin/bea/weblogic10/server -assignfile ../conf/price_assignment_set.xml -rolefile ../conf/price_role_set.xml -userfile ../conf/price_user_set.xml

$bash **bulkloader.sh** -apphome /<oracle_home>/j2ee/home -assignfile ../conf/price_assignment_set.xml -rolefile ../conf/price_role_set.xml -userfile ../conf/price_user_set.xml

The bulk loader will display error messages if problems occur. For more details, you can use the -verbose argument.

You can update Users and Roles with the bulk loader. The existing tables in the database will be overwritten. You cannot modify the Role Assignment table; however, you can add new Role Assignments.

User Management Security

In order to ensure the security of the application, the following security features are available in User Management:

- The AUTOCOMPLETE attribute in configurable on forms where passwords or user names are entered. By default, AUTOCOMPLETE is set to ON, so that sensitive information is stored.

  <ConfigRoot>/suite/suite.properties/suite.loginform.autocomplete = ON

- The session time out value is set in suite.httpsession.timeout. By default, it is set to 1800 seconds.

  <ConfigRoot>/suite/suite.properties/suite.httpsession.timeout = 1800

- The configure login time out value is independent of the session time out and should be of a shorter time period than the session time out. If configure time out value is not set, it defaults to the session time out value. By default, it is set to 1800 seconds.

  <ConfigRoot>/suite/suite.properties/suite.userlogin.timeout = 1800

- The attribute on the session ID cookie is set for secure deployments only so that the cookie can be transmitted using HTTPS and over an encrypted network. The default value is FALSE.

  <ConfigRoot>/suite/suite.properties/suite.cookie.secure = FALSE

- The application can be configured so that the logout page can either be displayed to the user or not. If the logout page is displayed, the user clicks **Login** to return to the Login page and **Close** to close the browser. The default setting is not to show the logout page but to return the user to the login page after logout.<ConfigRoot>/suite/suite.properties/suite.logoutpage.show = FALSE

Setting Up the Password Policies

Use the useraccount.properties file, located in <install-dir>/config/UserManagement, to set up the following password policies for the user accounts:

- Password expression and length
- Previous password check
- Password expiration period
- Maximum allowed unsuccessful login attempts

Note that password policies are enabled by default.

Setting Up the Access to Merchandise and Location Hierarchy

Use the usermanagement.properties file, located in <install-dir>/config/UserManagement, to specify the lowest merchandise and location hierarchy level accessible to the user accounts.

Enter an appropriate hierarchy levels in the **merchandizeMaxRoleAssignmentDepth** and **locationMaxRoleAssignmentDepth** fields.

Markdown Optimization Sample XML Files

This section provides sample input files for adding or updating users and roles.

User Sample XML File

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
- <user-set hash-passwords="true" xmls:xsi="http://www.w3.org/2001/XMLSchema-
   instance" xsi:noNamespaceSchemaLocation="user-set.xsd">
   <user username="view" password="view" last-name="Viewer" first-name="Joe"
    middle-initial="R" employeeID="1" title="El Presidente"/>
   <user username="submit" password="submit" last-name="Submitter" first-
    name="Jane" middle-initial="Y" employeeID="2" title="serf"/>
   <user username="approve" password="approve" last-name="Approver" first-
    name="Nancy" middle-initial="R" employeeID="3" title="El Presidente"/>
   <user username="titusten" password="titusten" last-name= "user" first-
    name="test" middle-initial="U" employeeID="4" title="serf"/>
   <user username="chain" password="chain" last-name="Franklin" first-
    name="Aretha" middle-initial="A" employeeID="5" title="Respect"/>
   <user username="brm_price" password="brm_price" last-name="ruler" first-
    name="business" middle-initial="P" employeeID="6" title="fool"/>
    <user username=”markdown_approve” password=”markdown_approve” last-
    name=”Approver” first-name=”Exception” middle-initial=”X” employeeID=”7”
    title=”Price Markdown Approver”/>
 </user-set>
 <!-- This XML supports adding/replacing "users" for the User Management
 subsystem. -->
-<!--
Note:
1. All role keys must be unique among all applications. Names like
PRICE_APPROVER, PLAN_EDITOR, and PLACE_READER are expected. They
must match those persisted into the DB.
2. The Users with a given username must be present in the DB prior to
this file being processed by the bulkloader.
3. The location and merchandise attributes are pipe delimited strings
of client load IDs. The first node is just below the root (Chain Level)
node. An empty attribute represents a chain level assignment.
4. All elements and attributes are case sensitive and all are lowercase.
5. The values of the Merchandise and Location hierarchy client load IDs are
based on Markdown Optimization dataset.
```

Roles Sample XML File

```
 <?xml version="1.0"encoding="UTF-8"?>
-<role-set xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="role-set.xsd">
 -<role key="PRICE_APPROVER">
    <action key="PRICE_APPROVE"/>
  </role>
 -<role key="PRICE_MARKDOWN_APPROVER">
    <action key="PRICE_MARKDOWN_APPROVE"/>
  </role>
 -<role key="PRICE_SUBMITTER">    <action key="PRICE_SUBMIT"/>
  </role>
 -<role key="PRICE_VIEWER">
    <action key="PRICE_VIEW"/>
  </role>
 -<role key="BRM_PRICE_VIEW">
    <action key="BRM_PRICE_VIEW"/>
  </role>
 -<role key="BRM_PRICE_EDIT">
    <action key="BRM_PRICE_EDIT"/>
  </role>
 -<role key="BRM_PROFITLOGIC_VIEW">
    <action key="BRM_PROFITLOGIC_VIEW"/>
  </role>
```

```
 -<role key="BRM_PROFITLOGIC_EDIT">
    <action key="BRM_PROFITLOGIC_EDIT"/>
  </role>
</role-set>
<!-- This XML supports adding/updating "roles" in the User Management
     subsystem. -->
-<!--
Note:
1. All role keys must be unique among all applications. Names like
PRICE_APPROVER, PLAN_EDITOR, and PLACE_READER are expected. They
must match those persisted into the DB.
2. The Users with a given username must be present in the DB prior to
this file being processed by the bulkloader.
3. The location and merchandise attributes are pipe delimited strings
of client load IDs. The first node is just below the root (Chain Level)
node. An empty attribute represents a chain level assignment.
4. All elements and attributes are case sensitive and all are lowercase.
5. The values of the Merchandise and Location hierarchy client load IDs are
based on Markdown Optimization dataset.
```

Role Assignment Sample XML File

```
Role Assignment Sample xml File (For MDO Config Guide)
<?xml version="1.0"encoding="UTF-8"?>
-<role-set xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="role-assignment-set.xsd">
<!-- This role guards the entire application -->

<role key="PRICE_USER">
<!-- Implicitly prevent "root" user from using Price -->
 <user-assignment username="approve">
  <node location="" merchandise=""/>
  </user-assignment>
 <user-assignment username="submit">
  <node location="" merchandise=""/>
  </user-assignment>
 <user-assignment username="view">
  <node location="" merchandise=""/>
 </user-assignment>
 <user-assignment username="titusten">
  <node location="" merchandise="1|1|11"/>
  <!-- worksheet 1, merchandise 85639-->
  <node location="" merchandise="1|1|12"/>
  <!-- worksheet 2, merchandise 86555-->
  <node location="" merchandise="1|1|14"/>
  <!-- worksheet 3, merchandise 87205-->
  <node location="" merchandise="1|1|15"/>
  <!-- worksheet 4, merchandise 87397-->
 </user-assignment>
 <user-assignment username="chain">
  <node location="" merchandise=""/>
 </user-assignment>
 <user-assignment username="markdown_approve">
  <node location="" merchandise=""/>
 </user-assignment>
</role>
```

```
<role key="PRICE_APPROVER">
 <user-assignment username="approve">
  <node location="" merchandise=""/>
 </user-assignment>
 <user-assignment username="titusten">
  <node location="" merchandise="1|1|11"/>
  <!-- worksheet 1, merchandise 85639-->
  <node location="" merchandise="1|1|12"/>
  <!-- worksheet 2, merchandise 86555-->
  <node location="" merchandise="1|1|14"/>
  <!-- worksheet 3, merchandise 87205-->
  <node location="" merchandise="1|1|15"/>
  <!-- worksheet 4, merchandise 87397-->
 </user-assignment>
 <user-assignment username="chain">
  <node location="" merchandise=""/>
 </user-assignment>
</role>

<role key="PRICE_SUBMITTER">
 <user-assignment username="submit">
  <node location="" merchandise=""/>
 </user-assignment>
 <user-assignment username="titusten">
  <node location="" merchandise="1|1|11"/>
  <!-- worksheet 1, merchandise 85639-->
  <node location="" merchandise="1|1|12"/>
  <!-- worksheet 2, merchandise 86555-->
  <node location="" merchandise="1|1|14"/>
  <!-- worksheet 3, merchandise 87205-->
  <node location="" merchandise="1|1|15"/>
  <!-- worksheet 4, merchandise 87397-->
</user-assignment>
<user-assignment username="markdown_approve">
 <node location="" merchandise=""/>
</user-assignment>
</role>

<role key="PRICE_VIEWER">
 <user-assignment username="view">
   <node location="" merchandise=""/>
 </user-assignment>
 <user-assignment username="titusten">
  <node location="" merchandise="1|1|11"/>
  <!-- worksheet 1, merchandise 85639-->
  <node location="" merchandise="1|1|12"/>
  <!-- worksheet 2, merchandise 86555-->
  <node location="" merchandise="1|1|14"/>
  <!-- worksheet 3, merchandise 87205-->
  <node location="" merchandise="1|1|15"/>
  <!-- worksheet 4, merchandise 87397-->
 </user-assignment>
</role>

<role key="BRM_PRICE_EDIT">
 <user-assignment username="root">
  <node location="" merchandise=""/>
 </user-assignment>
 <user-assignment username="chain">
  <node location="" merchandise=""/>
```

```
 </user-assignment>
 <user-assignment username="approve">
  <node location="" merchandise=""/>
 </user-assignment>
 <user-assignment username="submit">
  <node location="" merchandise=""/>
 </user-assignment>
 <user-assignment username="view">
  <node location="" merchandise=""/>
 </user-assignment>
 <user-assignment username="titusten">
  <node location="" merchandise=""/>
 </user-assignment>
 <user-assignment username="brm_price">
  <node location="" merchandise=""/>
 </user-assignment>
</role>

<role key="BRM_PROFITLOGIC_EDIT">
 <user-assignment username="root">
  <node location="" merchandise=""/>
 </user-assignment>
 <user-assignment username="chain">
  <node location="" merchandise=""/>
 </user-assignment>
<user-assignment username="approve">
 <node location="" merchandise=""/>
</user-assignment>
<user-assignment username="submit">
 <node location="" merchandise=""/>
</user-assignment>
<user-assignment username="view">
 <node location="" merchandise=""/>
</user-assignment>
<user-assignment username="titusten">
 <node location="" merchandise=""/>
</user-assignment>
</role>
</role-assignment-set>

Note:
1. All role keys must be unique among all applications. Names like
PRICE_APPROVER, PLAN_EDITOR, and PLACE_READER are expected. They
must match those persisted into the DB.
2. The Users with a given username must be present in the DB prior to
this file being processed by the bulkloader.
3. The location and merchandise attributes are pipe delimited strings
of client load IDs. The first node is just below the root (Chain Level)
node. An empty attribute represents a chain level assignment.
4. All elements and attributes are case sensitive and all are lowercase.
5. The values of the Merchandise and Location hierarchy client load IDs are
based on Markdown Optimization dataset.
```

3

Business Rule Property Manager

This chapter explains how to configure the business rules using the Business Rule Property Manager.

It contains the following sections:

Introduction

The Business Rule Property Manager (BRPM) is a Markdown Optimization utility that is used to view and change business rule settings. Business rules determine which data is used by the application for an optimization. In effect, business rules specify retailer constraints that are used by the application to determine markdowns and forecasting.

The application provides a file that contains the business rule definitions. The business rule definitions specify the constraints that apply to business rule instances (mappings between location and merchandise hierarchy levels and business rule values). The definitions are configurable; however, most of the business rules have default values that can be used to perform any initial application work.

The Markdown Optimization business rules are implemented through the inference rules, using values managed in the BRPM. Both the inference rules and the business rules are points of customization for the application.

The BRPM is accessed through the application Main Menu. A user's ability to view and change business rule settings is specified by the permissions attached to the user role(s) assigned to them. These roles are assigned using the User Management utility.

For more information, see the *Oracle Retail Markdown Optimization Administration Guide*. The actions used by BRPM roles are defined in the business rule definition file (discussed later in this chapter).

The BRPM is used to:

- view current business rule settings for specific items
- change business rule settings in time for the next optimization
- change business rule settings when problems occur during a model run, so that the problem can be fixed and the model run restarted
- view the history of business rule changes

Getting Started

Once you have completed the initial installation and configuration of Markdown Optimization, you must load all the data required by the application, in a format specified by the standard interface specifications and using the standard load procedure. (See the *Oracle Retail Markdown Optimization Operations Guide* for information about the standard load.) You can then configure the application to match the retailer's specific business requirements.

The model run updates the forecasts, recommendations, and metrics that are displayed in the Markdown Optimization application through the UI. You can perform an initial model run using the default values provided with the application. This will allow you to get the system up and running. It will also provided you with a baseline configuration that you can use when planning your advanced configuration.

The advanced configuration is necessary in order to obtain meaningful markdown recommendations and forecasts from Markdown Optimization.

In order to do a model run, you must configure the business rule definitions and load them into Markdown Optimization. The default business rule definitions are contained in /modules/tools/conf/DefaultRules/rule_definitions.xml. An editable copy of the business rule definitions can be found in config/businessrulemgr/rule_definitions.xml. Once you have edited this file, you can use /modules/tools/bin/brmadmin.sh to load the file into the application.

The default settings, are, in general, sufficient for an initial model run. These default values are set at the highest level for everything in the system. The exceptions are Out Dates and Planned Start Dates, which are installed without default values assigned. Prior to the model run, you should enter Out Dates, using either the BRPM or through the application UI (Maintaining Merchandise). If you are going to use Planned Start Date as your Model Start Date, you should enter that value as well, using either the BRPM or through the application UI (Maintaining Merchandise). Note that set-plannedstartdate in p4pgui-config.xml must be set to true in order to configure the Planned Start Date using the application UI. Excluded days for the planned start date can also be configured in p4pgui-config.xml.

For more information on the model run, see the *Oracle Retail Markdown Optimization Operations Guide*.

Default Business Rules

Markdown Optimization is configured, by default, with 16 default business rules accessible through the BRPM. The values for the business rules are fetched by the IR_BUSINESS_POLICY inference rule and used by that inference rule as well as others. The default business rules are, in effect, a subset of the default set of inference rules.

Certain of the default rules are only used by administrators for system-level configuration.

The default Business Rules are:

Table 3–1 Default Business Rules

Business Rule Name/ Display Name Used in the UI	Business Rule Description	Default Value
NO_TOUCH_AFTER_LAND No Touch 1st	The minimum number of weeks after the model start date before the item is eligible for a markdown. Note that "touch" refers to retail activity and "land" refers to the period when there are enough weeks of sales to start forecasting.	7
NO_TOUCH_AFTER_MKDN No Touch Between	The minimum number of weeks between markdowns (after the first one). Note that 0 is not a valid value.	7
MAX_MKDN_NO Max #	The maximum number of markdowns permitted for an item during its entire life cycle.	3
MIN_FIRST_MKDN Min Initial	The minimum amount for the first markdown, which is the lowest percentage drop allowed from the current ticket price at the time of the initial markdown.	0
MIN_OTHER_MKDN Min Other	The minimum amount for any markdown after the first one. The lowest percentage drop allowed from the current ticket price at the time of the markdown.	0
MAX_FIRST_MKDN Max Initial	The maximum amount for the first markdown, which is the highest percentage drop allowed from the current ticket price at the time of the initial markdown.	1
MAX_OTHER_MKDN Max Other	The maximum amount for any markdown after the first one. The highest percentage drop allowed from the current ticket price at the time of the markdown.	1
PLANNED_START_DT Start Date	The date when an item will first be sold.	-

Table 3–1 (Cont.) Default Business Rules

Business Rule Name/ Display Name Used in the UI	Business Rule Description	Default Value
OUT_DT Out Date	The date planned for the end of inventory or by which a specific sell-through target is to be reached.	-
INVENTORY_TARGET Sell Thru %	The planned percentage of sell-through for an item at the outdate. Expressed as a value between 0 and 1.	1
SALVAGE_WITHIN_TARGET Salv Within	The salvage value of remaining items if the target inventory is met. This is a percentage of the full price. Expressed as a value between 0 and 1.	1
SALVAGE_ABOVE_TARGET Salv Above	The salvage value of the remaining items if the target inventory is above the expected amount. This is a percentage of the full price. Expressed as a value between 0 and 1.	0
NO_TOUCH_BEFORE_OUT (Administrative Business Rule) No Touch EOL	The number of weeks before the outdate when markdowns are no longer permitted.	14
MIN_MKDN_FROM_FULL (Administrative Business Rule) Min % of Full	The minimum markdown, expressed as a percentage of the original full retail price. Used to narrow the list of possible prices for optimization.	0
MAX_MKDN_FROM_FULL (Administrative Business Rule) Max from Full	The maximum markdown, expressed as a percentage of the original full retail price. Used to narrow the list of possible prices for optimization.	1
MKDN_DAY_OF_WEEK Administrative Business Rule) Day of Week	A global setting for the day of the week on which markdowns occur. (Sunday = 1, Monday = 2, Tuesday = 3,....)	2
TEMP_MARKDOWNS_BLOCK	Defines whether TEMP markdowns are counted during the enforcement of the MinMarkdownInterval and MaxNumber- Markdowns (IR_ BUSINESS_POLICY). When value is 1, TEMP markdowns count.	1

Table 3–1 (Cont.) Default Business Rules

Business Rule Name/ Display Name Used in the UI	Business Rule Description	Default Value
POS_MARKDOWNS_BLOCK	Defines whether POS markdowns are counted during the enforcement of the MinMarkdownInterval and MaxNumber-Markdowns (IR_BUSINESS_POLICY). When value is 1, POS markdowns count.	1
MSD_FORCED_START_DT Forced Model Start Date	This is only used if the MODEL_START_OPTION in IR_MODEL_START_ OPTION is set to sellThrough. See the Inference Rule chapter for more information. The value is an override date that forces the Model Start Date to be the first fiscal day of the week of the Forced Model Start Date.	NONE
MSD_SELLTHROUGH_PCT Model Start Sell Through Pct	This is only used if the MODEL_START_OPTION in IR_MODEL_START_ OPTION is set to sellThrough. See the Inference Rule chapter for more information. The value is a threshold that triggers the assignment based on a ratio of sold units to total inventory.	2.00%
MSD_MAX_DELAY_WKS Max Model Start Delay	This is only used if the MODEL_START_OPTION in IR_MODEL_START_ OPTION is set to sellThrough. See the Inference Rule chapter for more information. The value is the maximum number of weeks to wait before automatically assigning the Model Start Date.	2

Table 3–1 (Cont.) Default Business Rules

Business Rule Name/ Display Name Used in the UI	Business Rule Description	Default Value
OUT_WKS Out Weeks	This provides an alternate calculation for an item's exit date if the business rule OUT_DT is not set. This is useful for the seasonal maintenance of exit dates. If an item's OUT_DT is not set, the alternate out date is calculated by combining the item's model_start_date value, the OUT_WKS value, and the global parameter OutDay. OUT_WKS provides the number of weeks that is added to an item's model start date, to determine a default out/exit week. Applying the global OutDay to it gives the specific alternate out date.	10

Business Rule Definitions

You may want to configure the business rules to meet the needs of your business. The sample file (rule_definitions.xml), located in /modules/tools/conf/SampleRules, provides an illustration of a set of business rules, including a configured attribute for Season Codes and some test rules that illustrate validation constraints. You can use this file as an advanced example of some possible approaches to take when planning your own configuration. However, your customization should be based on the default business rules. An editable copy of the business rule definition can be found in config/businessrulemgr/rule_definitions.xml. Once you have edited this file, you can use /modules/tools/bin/brmadmin.sh to reload the file in order to implement the changes you have made.

The XML schema for the business rule definitions file is located in tools/brmadmin/conf/brm_config.xsd.

Here is a sample business rule definition, including two attributes, taken from /modules/tools/conf/SampleRules/rule_definitions.xml:

```
<AttributeInfo name="SEASON_CODE"
   table="ITEMS_TBL"
   shortDescription="brm.rules.attribute.attr1.label"
   longDescription="brm.rules.attribute.attr1.description"
   allowOtherValues="N"/>
<AttributeInfo name="VENDOR"
   table="ITEMS_TBL"
   shortDescription="brm.rules.attribute.attr2.label"
   longDescription="brm.rules.attribute.attr2.description"
   allowOtherValues="Y"/>
<RuleDefinition name="MIN_FIRST_MKDN"
   shortDescription="brm.rules.params.minmarkdown.label"
   longDescription="brm.rules.params.minfirstmarkdown.description"
   readAction="BRM_PRICE_VIEW"
   editAction="BRM_PRICE_EDIT"
  <KeyLevel merchandiseLevel="DEFAULTLEVEL"
     locationLevel="DEFAULTLEVEL"
```

```
      matchAttribute1="N"
      matchAttribute2="N"/>
   <KeyLevel merchandiseLevel="WORKSHEET"
      locationLevel="WORKSHEET"
      matchAttribute1="N"
      matchAttribute2="N"/>
   <KeyLevel merchandiseLevel="WORKSHEET"
      locationLevel="WORKSHEET"
      matchAttribute1="N"
      matchAttribute2="Y"/>
   <ValueDefinition valueType="FLOAT"
      validationType="RANGE"
      shortDescription="brm.rules.value.markdownpct.label"
      longDescription="brm.rules.value.markdownpct.description"
      allowNullValues="N"
      defaultValue="0">
    <value ruleValue="0"/>
    <value ruleValue="1"/>
</RuleDefinition>
```

Each business rule definition contains the following information:

- The name of the business rule, in this case MIN_FIRST_MKDN.
- The short description resource ID for the business rule's name, which is displayed in the UI.
- The long description resource ID for the business rule description, which is displayed when a user hovers over the name in the UI.
- The read action and the write action associated with the business rule. Roles, which are assigned to specific users and determine their permissions, are made up of actions. In order for users to be able to view and/or edit a business rule in the UI, they must be assigned a role that includes some combination of the following actions at the desired level or higher:
  - PRICE_VIEW
  - PRICE_SUBMIT
  - BRM_PRICE_VIEW
  - BRM_PRICE_EDIT

  In addition, in order to be able to view and/or edit administrative business rules, users must be assigned a role that includes:
  - BRM_PROFITLOGIC_VIEW
  - BRM_PROFITLOGIC_EDIT

  For more information on actions and roles, see the Markdown Optimization Online Help.
- An arbitrary number of key levels, which specify at what levels an instance of the business rule can be matched to an item. Each key level contains a merchandise hierarchy level, a location hierarchy level, and optional custom attributes that are used to determine the match between an item and a rule. To determine the rule mapping, matching occurs in the following order of precedence:
  1. Search the merchandise hierarchy from low to high for a match.
  2. Search the location hierarchy from low to high for a match.
  3. If an attribute is set to Y, match that item's value.

4. If a attribute is set to N, match any attribute value.

   If a rule is set at more than one level (for example outdates at both the merchandise/location level and the merchandise/location/attribute level), matching occurs at whatever the lowest level is, given the circumstances.

   For the example rule definition shown above, matching of rule to item occurs at the DEFAULTLEVEL DEFAULTLEVEL level with any attribute, at the Worksheet Worksheet level with any attribute, and at the Worksheet Worksheet level with the Vendor attribute.

- The type of value for the rule:
  - Integer
  - Floating point number
  - Date
  - String
- Validation, by range, enumeration, or none. If range, then the minimum and maximum values are given. If enumeration, a list of values is provided.
- Whether or not null values are allowed.
- The default value for the rule. If no default value is assigned, then NULL is assumed.
- If range is being used for validation, in combination with a valid type, the minimum and maximum values of the range are provided.

Loading Business Rule Definitions

When you first begin using the application and whenever you make changes, you must load the business rule definitions file into the database, using brmadmin.sh.

Here is the usage for the brmadmin.sh script.

Server Mode (the default), which sends the request to the application server:

brmadmin.sh [-server] <*config_root*> <*rule_definitions*> *[<host> <port>]*

Client Mode, which processes the request on the client side:

brmadmin.sh [-client] <*config_root*> <*rule_definitions*>

where

- <config_root> – the root directory of the application configuration files.
- <rule_definitions> – the name of the XML file that contains the rule definitions.
- <host> - the application server host
- <port> - the application server port
- -h - displays help message
- -p - disables execution of database load procedures

You must preserve business rule definitions required by the application as well as those required by any inference rules that you have customized.

Business rule instances are affected when you modify business rule definitions. If you change rule value types, business rule instances may be deleted. In addition, changes

to definitions may cause inconsistencies between the rules and the instances. As a result, the application may not perform properly.

If you change business rule definitions or add new ones, you may have to modify the grid configuration for the BRPM (see "Business Rule Property Manager Grid Configuration" on page 3-13).

Configuring Business Rule Definitions

When configuring business rules to meet business needs, consider the following:

- When configuring key levels, you must manage the settable levels in conjunction with the inheritance hierarchy and user access.
- Since the application business rules are implemented through the inference rules, changes to the ir.sql may affect rule instances.
- Editing business rule definitions to change validations or default values may affect rule instances.
- Editing business rule definitions to change validations or default values may affect system performance.
- If you add a new business rule or change an existing one, you may need to add resources or modify the grid configuration.

Business Rule Instances

A business rule instance is a specific mapping between a key and a rule value. When BRPM is installed, instances for the business rules exist at the top level and have the default values assigned to them (even if the top level is not a settable key level as defined in the business rule definition). If a business rule instance is deleted, the object that was assigned that instance will then inherit the settings of the instance at the next higher precedence level in the hierarchy. If the top level is deleted, the instance returns to the default value in the business rule definition file.

Guidelines for Entering Business Rule Instances

You can enter values for business rules either by using the BRPM application or the BRPM API. Both methods validate the instance against the BRPM rule definitions. When using the BRPM, you must be assigned a role that permits you to make changes to business rule values. For more information on Roles, see the application Online Help.

In addition, you can enter some item-level values through the Markdown Optimization Maintaining Merchandise screen. For more information, see the *Oracle Retail Markdown Optimization User Guide*.

Business rule instances must be consistent with business rule definitions:

- Instances must be settable at the desired level, as defined in the rule definitions.
- Instances must conform to the validations defined in the rule definitions, which include the value type.
- Each instance must have an associated business rule definition.
- The key level of each instance must be permitted by the rule definition.
- The attribute values used in the instance keys should be consistent with the attributes in the BRPM configuration.

You can use the BRPM to view a business rule value that was in effect for a particular date. The UI displays all the rule values that would apply using inheritance. The value on the target date is the one with the highest precedence. For more information, see the *Oracle Retail Markdown Optimization User Guide*.

Custom Attributes

Attributes are optional variables that can be added to a specific business rule definition. Two attributes are permitted. Attributes extend the business rule key and are used to determine the match between a rule and an item. Custom attributes should be added to the rule_definitions.xml file.

The attribute definition includes:

- the attribute name, which must be consistent with the column name in the source table.
- the name of the table that includes the column used for the attribute name. The following tables can be used:
  - ITEMS_TBL
  - ITEMS_CDA_TBL
  - MERCHANDISE_HIERARCHY_TBL
  - MERCH_ATTR_TBL
  - LOCATION_HIERARCHY_TBL
  - LOCATION_ATTR_TBL
- the resource ID for the attribute's name, which is displayed in the UI.
- the resource ID for the attribute description, which is displayed when a user hovers over the name in the UI.
- whether an attribute value other than one from the current set of values is valid.

To configure custom attributes (for example, Season Code and Vendor), you should define the resources used for their display as part of businessrulemgrResources.properties:

```
# Rules grid - Attributes
brm.rules.attribute.group.label=Attributes
brm.rules.attribute.group.description=Attributes
brm.rules.attribute.attr1.label=Season
brm.rules.attribute.attr1.description=Season Code
brm.rules.attribute.attr2.label=Vendor
brm.rules.attribute.attr2.description=Vendor
```

Once the custom attributes have been defined, you must run com.profitlogic.db.birch.LoadBRMAttributeValues (part of PRERUN) after you run brmadmin.sh in order to see the custom attributes changes in the Markdown Optimization application. LoadBRMAttributes loads values into BRM_ATTRIBUTE_VALUE_TBL. The application derives the values for the attributes displayed on the BRM page from this table.

Business Rules and Inference Rules

The Markdown Optimization business rules are implemented through the inference rules (discussed later in this chapter), using values customized and set in the BRPM.

The IR_BUSINESS_POLICY inference rule (and other inference rules that require the data) can obtain business rule values in two ways:

- The getBRValue function obtains the current value, including any values that have changed since the last pre-model run step.
- During the pre-model run stage, the item_brm_rules table is populated at the item level to provide quick access to business rule values during the model run.

You can configure IR_BUSINESS_POLICY. For example:

- If you change the name of the business rule in the BRM, you should also change it in IR_BUSINESS_POLICY.
- You can define a business rule value as a constant, if the value does not vary by merchandise level, location level, or attribute, by defining the constant in IR_BUSINESS_POLICY.

Business Rule Property Manager Bulk Loader

The BRPM Bulk Loader provides a means for staging and loading a set of business rule instances. This utility is included within the standard interface and standard load (for more information, see the *Oracle Retail Markdown Optimization Operations Guide*), but can also be implemented separately if new or updated business rule instances need to be loaded outside the normal scheduled batch processes. The Bulk Loader validates the business rule instances according to the guidelines described in "Guidelines for Entering Business Rule Instances" on page 3-9.

Business Rule Instances Standard Interface Specification (ASH_BRM_INSTANCE_TBL)

The data to be loaded by the Business Rule Property Manager bulk loader utility must conform to the following standard interface specification.

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).

Table 3–2 Business Rule Instances Standard Interface Specification

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
MERCHANDISE_KEY	Key for this level of the hierarchy.	String	50	N
MERCHANDISE_LEVEL	ID for this level of the hierarchy.	String	50	N
LOCATION_KEY	Key for this level of the hierarchy.	String	50	N
LOCATION_LEVEL	ID for this level of the hierarchy.	String	50	N
RULE_NAME	The name of the business rule associated with the item.	String	64	N
RULE_VALUE	The business rule value assigned to the item.	StringValues < 1 should be expressed as 0.n.	100	N

Table 3–2 (Cont.) Business Rule Instances Standard Interface Specification

Attribute	Attribute Description	Data Type	Maximum Length	Nullable Y/N
ATTRIB1_VALUE	The specific value associated with the item for custom attribute 1.	String	100	Y
ATTRIB2_VALUE	The specific value associated with the item for custom attribute 2.	String	100	Y
DELETE_FLAG	A flag to indicate whether the instance is to be deleted or inserted. 0 = insert (the default). 1 = delete.	Integer	1	N

Loading Instances

The Standard Load scripts that stage and load the data into the application stage and load business rule instances. In order to invoke the BRPM Bulk Loader utility separately, as a manual process, do the following:

bash **pl_stage_file.sh** --controldir=<*directory with control files*> --logdir=<*log output directory*> <*file containing standard interface-compliant BRM rule instances*>

bash **pl_load_data.sh** --logdir=<*log output directory*> "com.profitlogic.db.birch.LoadBRInstances"

The utility validates whether or not the instance key is a legal key at the specified level and whether the instance value is a legal value, as specified in the definition. If the validation fails, the procedure terminates and no changes are made.

Business rule definitions contained in config/businessrulemgr/rule_definitions.xml are loaded using brmadmin.sh.

Business Rule Property Manager Properties

BRPM properties may need to be configured prior to the deployment of the application. The properties are located in configroot/businessrulemgr/businessrulemgr.properties. The settings in this file can be overwritten by retailer settings.

Table 3–3 Business Rule Property Manager Properties

Property	Description	Default Value
numBrowsableMerchLevels	The number of merchandise hierarchy levels that can be browsed in the BRPM UI.	4
numBrowsableLocLevels	The number of location hierarchy levels that can be browsed in the BRPM UI.	2
numFindableMerchLevels	The number of additional merchandise hierarchy levels that can be accessed using the BRPM find feature.	2
numFindableLocLevels	The number of additional location hierarchy levels that can be accessed using the BRPM find feature.	1
numExpandableMerchLevels	The number of levels that the merchandise hierarchy can be expanded to in the BRPM UI.	4
numExpandableLocLevels	The number of levels that the location hierarchy can be expanded to in the BRPM UI.	3

Guidelines for Setting BRPM Properties

Use the following guidelines in planning the configuration of the BRPM properties:

- The number of browsable merchandise hierarchy levels should equal the application worksheet merchandise hierarchy levels.
- The number of browsable location hierarchy levels should equal the application worksheet location hierarchy levels.
- The number of findable merchandise hierarchy levels should equal (the total number of merchandise levels – the number of browsable merchandise hierarchy levels).
- The number of findable location hierarchy levels should equal (the total number of location hierarchy levels – the number of browsable location hierarchy levels).
- The number of expandable merchandise hierarchy levels should equal the number of browsable merchandise hierarchy levels.
- The number of expandable location hierarchy levels should equal the number of browsable location hierarchy levels.

In addition, keep in mind that:

- the BRPM validates that the total number of levels defined in the properties file does not exceed the number of levels defined in the database.
- to forestall performance or memory problems, set the number of levels in the properties file close to Class in the merchandise hierarchy.
- the default values for common.hierarchy.cache.timeout.hour in configroot/suite/suite.properties may need to be configured.

Business Rule Property Manager Grid Configuration

For business rules such as OUT_DT that allow null values, a custom property must be added to the grid configuration. For more information on configuring the front end, see and the Front End Configuration chapters of this book.

```
<column-def>
   <key>OUT_DT</key>
   <column-def-properties type="date" display-type="date" db-column-name="name''
db-table-name="name" editable="true" sortable="true" orderable="true"
hideable="true"
groupId="GROUP_HEADER" visibility="never-visible"/>
   <custom-property name="convertZeroToNull" value="true" custom-type="display"/>
->  <custom-property name="allowNone" value="true" custom-type="display"/>
   </column-def>
```

Example Configuration

The INVENTORY_TARGET business rule is configured to express the target inventory level as a percentage of sell-through. To change the inventory target so that it is expressed as the number of end inventory units, complete the following process.

1. In the INVENTORY_TARGET business rule, configure the value Type as INT.

```
<RuleDefinition name="INVENTORY_TARGET"
  shortDescription="brm.rules.params.inventorytarget.label"
  longDescription="brm.rules.params.inventorytarget.description"
  readAction="BRM_PRICE_VIEW"
  editAction="BRM_PRICE_EDIT"
```

```
    <KeyLevel merchandiseLevel="DEFAULTLEVEL" locationLevel="DEFAULTLEVEL"
    matchAttribute1="N" matchAttribute2="N"/>
   <KeyLevel merchandiseLevel="WORKSHEET" locationLevel="WORKSHEET"
    matchAttribute1="N" matchAttribute2="N"/>
   <KeyLevel merchandiseLevel="OPTIMIZATION" locationLevel="OPTIMIZATION"
    matchAttribute1="N" matchAttribute2="N"/>
   <ValueDefinition valueType="INT" validationType="RANGE"
    shortDescription="brm.rules.value.inventorytarget.label"
    longDescription="brm.rules.value.inventorytargetdescription"
    allowNullValues="N"
    defaultValue="0">
     <Value ruleValue="0"/>
     <Value ruleValue="1000000000"/>
   </ValueDefinition>
<RuleDefinition>
```

2. To make INVENTORY_TARGET an integer in the grid, set the BRPM grid configuration in configroot/businessrulemgr/client/grids/brm-column-list.xml as follows:

```
<column-def>
  <key>INVENTORY_TARGET</key>
  <column-def-properties type="integer" display-type="integer"
    db-column-name="name" db-table-name="name" editable="true"
    sortable="true" orderable="true" hideable="true" groupid="GROUP_HEADER"
visibility="never-visible"/>
  <custom-property name="convertZeroToNull" value="true"
    custom-type="display"/>
<column-def>
```

3. Change the resources file so that the display label and the rule

   description reflect the change from Sell Through Percent to

   Ending Inventory Units. The resources file is located in configroot/suite/resources/businessrulemgrResources.properties.

```
brm.rules.params.inventorytarget.label = End Inv Units
brm.rules.params.inventorytarget.description = The inventory target at the out
date as ending inventory units
```

4. Specify the following in p4pgui-config.xml:

```
<merchandise-maint-params endingInv-input-type="endingInvUnits">
```

5. Change the default configurations for INT_MOD_INV_TARGET_ST_PERC and INT_MOD_INV_TARGET_END_UNITS in the custom column file by commenting out or removing default definitions and un-commenting the alternative definitions for those columns.
6. Make the column INT_MOD_INV_TARGET_ST_PERC un-editable, and make the column INT_MOD_INV_TARGET_END_UNITS editable in the maintenance grid configuration files, p4p-maint-grid.xml and p4p-maint-grid-groups.xml.
7. Make the column INT_MOD_INV_TARGET_ST_PERC un-editable, and make the column INT_MOD_INV_TARGET_END_UNITS editable in the maintenance grid configuration file, p4p-maint-grid-flat.xml.
8. Edit and re-apply the ir.sql file. Inventory target in units is calculated for each item by the inventoryTarget column of ir_business_policy (and ir_business-policy_c). By default, it calculates a value using the sell through percent obtained from the BRPM. Change the code to use the value directly as the number of units:

```
TO_NUMBER(
  getBRValue('INVENTORY_TARGET',
    i.merchandise_id, i.location_id,
    brm_attribute1, brm_attribute2))
as inventorytarget,
```

4

Inference Rules

Inference rules, which are customized for specific retailer configurations, are used by the MDO model run.

This chapter contains the following sections:

Introduction

Inference rules define queries specifying particular views into the database that provide customization points for Markdown Optimization. An inference rule corresponds to a specific business policy. For example, views define relevant dates and data, the values needed for model runs, and which metrics to calculate and populate in the tables visible through the UI.

Inference rules define the interface between the data and the model. All data that is passed to the model is controlled by inference rules. In addition, much of the data that is passed to the ITEM_DATA table and the UI is also controlled by inference rules. (However, some data is passed to the UI directly through the load statements.)

Markdown Optimization is installed with default inference rules, provided in the **ir.sql** file, which is located in **config/db.config**. The ir.sql file is overwritten during every subsequent installation. If you are going to customize **ir.sql**, it is recommended that you create a copy of the changes in **config/db.config**. Keeping a copy of your customization can be helpful in troubleshooting. In addition, this will allow you to apply your changes to any upgrade, which is important, as the default inference rules can change between release of the application.

Two scripts are available that you can use to apply the **ir.sql** to the database schema:

- **plconfiguredb.sh**, used by the installer
- **configdb.sh**, located in **config/db.config**

For example:

$ **bash configdb.sh** *dbalias username password ir.sql*

You can use **configdb.sh** to apply your **custom_ir.sql** to the database.

Note that certain inference rule values can be managed using the Business Rule Property Manager (BRPM).

Inference Rule Access

To obtain the best performance, you can configure how the Calculation Engine queries inference rules.

Inference Rules can be accessed in three different ways. The way inference rules are accessed is customizable and can impact performance. It is possible to process more than one item at a time; that is, it is possible to have fewer, larger queries.

The inference rule access strategy is configured in **delphi.properties**, as follows:

Table 4–1 Inference Rule Access Configuration Settings

Configuration Setting	Form of Where Clause in Query
strategy.activitydata=single	where item_id = 1234
strategy.activitydata=list	where item_id in (1234, 5678, ...n), where n is a value between 1,000 and 10,000.
strategy.activitydata=temptable	where item_id in (select from temp_table)

The access level can be configured for the following Inference Rules, which are a direct interface to the Calculation Engine:

Table 4–2 Inference Rules Strategy Setting Names

Inference Rule Name	Strategy Setting Name
IR_ACTIVITY_DATA	activitydata
IR_BUSINESS_POLICY	businesspolicy
IR_FORCED_MARKDOWNS	forcedmarkdowns
IR_ITEM_DATES	itemdates
IR_ITEM_PARAMETERS	itemparameters
IR_ITEM_PRICES	itemprices
IR_MARKDOWN_CALENDAR	markdowncalendar
IR_MODEL_VALUES	modelvalues
IR_PAST_TICKET_PRICES	pastticketprices
IR_PENDING_MARKDOWNS	pendingmarkdowns
IR_PLANNED_PROMOS	plannedpromos
IR_PRICE_LADDER	priceladder
IR_PRIOR_DISTRIBUTION	distribution

Most inference rules have a default strategy option of *list*. Here is an example of override settings for each of the inference rules listed in Table 4–3, " Inference Rules" that can be used in **delphi.properties** (see Table 4–1, " Inference Rule Access Configuration Settings").

strategy.activitydata=temptable

strategy.businesspolicy=list

strategy.forcedmarkdowns=list

strategy.itemdata=list

strategy.itemparameters=list

strategy.itemprices=list

strategy.markdowncalendar=list

strategy.modelvalues=list

strategy.pastticketprices=list

strategy.pendingmarkdowns=list

strategy.plannedpromos=list

strategy.priceladder=list

strategy.distribution=single

Note that some inference rules have interdependencies that can impact performance.

Performance Tuning Recommendations

If you make changes to inference rules and performance during an model run or a What If simulation becomes slow, consider the following:

1. Check to see if that the Database is fully loaded and that the CPU is being fully utilized.
2. Use a Database Monitoring software tool such as TOAD to check the active sessions in the database.
3. Typically, n number of connections are visible in the database. Are all the connections sitting on the same query? If so, the query is a bottleneck for the throughput of the run.
4. If the query is accessing an inference rule using an access strategy of list or temptable and is taking too long, try changing the access strategy.

   Note that the single access strategy will provide reasonable but not optimum performance. The list and temptable strategies are recommended for optimum performance.

Inference Rule Categories

Inference rules can be divided into two general categories:

- Inference rules that provide information used in the model run. Item data, business constraints, and model parameters. Some of these generally require customization and others do not.
- Inference rules that produce metrics, some of which are displayed in the UI.

In some cases, two versions of an inference rule exist: IR_NAME and IR_NAME_C. The IR_NAME form is used when a item is optimized individually and the IR_NAME_C form is used when the item is optimized as part of a pricing group. In certain cases, pricing groups require special behavior and the inferences rules provide the means to accomplish this (for example, to align outdates in IR_ITEM_DATES_C). Some IR_NAME_C inference rules contain a COLLECTION_ID.

This section describes a subset of the Markdown Optimization inference rules. Note that some inference rules are included in more than one category.

Table 4–3 Inference Rules

Inference Rule Name	Discussed On...
Inference Rules that are part of the basic configuration and that are typically customized.	
IR_ITEM_DATES and IR_ITEM_DATES_C	on page 4-13
IR_ITEM_PRICES and IR_ITEM_PRICES_C	on page 4-15
IR_MODEL_START	on page 4-16
IR_MODEL_START_OPTION	on page 4-16
IR_PENDING_MARKDOWNS	on page 4-19
IR_SEASONALITY_ATTRIBUTE	on page 4-22
Inference rules that are part of business policies and that are typically customized.	
IR_BUSINESS_POLICY and IR_BUSINESS_POLICY_C	on page 4-6
IR_BLOCKED_MARKDOWN and IR_BLOCKED_ MARKDOWN_C	on page 4-6
IR_COLLECTION_INFO	on page 4-8
IR_MARKDOWN_CALENDAR	on page 4-15
IR_MARKDOWN_CALENDAR_EX and IR_ MARKDOWN_CALENDAR_EX_C	on page 4-16
IR_PLANNED_PROMOS	on page 4-20
IR_PRICE_LADDER and IR_PRICE_LADDER_C	on page 4-21
Inference rules that are provided by Analytical Services.	
IR_ITEM_BASE_DEMAND	on page 4-11
IR_ITEM_PARAMETERS	on page 4-14
IR_MODEL_VALUES	on page 4-17
IR_PLANNED_PROMOS	on page 4-20
IR_STATE_TRANS_CONFIG_OVERRIDE	on page 4-22
IR_STATE_TRANS_PREV_RUN	on page 4-22
Inference rules that are typically not changed.	
IR_ACTIVITY_DATA	on page 4-6
IR_ELIGIBLE	on page 4-10
IR_PAST_TICKET_PRICES	on page 4-19
IR_ITEM_IDS and IR_ITEM_IDS_C	on page 4-14
Inference rules that make information available to the UI and that are used during the KPI calculations. These inference rules populate the ITEM_DATA table.	
IR_COLLECTION_INFO	on page 4-8
IR_FRONT_END_IDS	on page 4-11
IR_ITEM_DATES	on page 4-13
IR_ITEM_PRICES	on page 4-15
IR_WAREHOUSE	on page 4-23
IR_USER_TEXTS	on page 4-23

Table 4–3 (Cont.) Inference Rules

Inference Rule Name	Discussed On...
IR_USER_DATES	on page 4-23
IR_USER_FLOATS	on page 4-23
IR_WORKSHEET_IDS	on page 4-24
Inference rules that use forecast and markdown recommendation information and that populate the ITEM_DATA table.	
IR_FORECAST_METRICS	on page 4-10
IR_O_USER_TEXTS	on page 4-18
IR_O_USER_DATES	on page 4-17
IR_O_USER_FLOATS	on page 4-18
IR_PROJ_MKDNS	on page 4-21
Inference rules used to configure Pricing Groups.	
IR_ITEM_COLLECTION	on page 4-11
IR_ITEM_COLLECTION_OPTION	on page 4-12
Inference rules used by What If.	
WIF_FORECAST_DATA	on page 4-23
Additional inference rules.	
IR_DISPLAY_PROMOS	on page 4-9
IR_FE_WAREHOUSE	on page 4-10
IR_FORCED_MARKDOWNS	on page 4-10
IR_FORECAST_METRICS_POSTRUN	on page 4-10
IR_HISTORIC_METRICS	on page 4-11
IR_ITEM_DAILY_WEIGHTS_OVERRIDE	on page 4-12
IR_ITEM_INFO and IR_ITEM_INFO_C	on page 4-14
IR_LOCATION_HIERARCHY	on page 4-15
IR_LOC_OPT_LEVEL	on page 4-15
IR_MERCH_OPT_LEVEL	on page 4-16
IR_MERCHANDISE_HIERARCHY	on page 4-16
IR_METRICS	on page 4-16
IR_MISSING_WEEKS	on page 4-16
IR_P4P_ITEMS_CONFIG	on page 4-18
IR_PRIOR_DISTRIBUTION	on page 4-21
IR_PROCESS_NULL_OUTDT	on page 4-21
IR_ROLLUPS	on page 4-22
IR_SEASON_METRICS	on page 4-22
IR_WIF_PROJ_OH_UNITS_EFF_DT	on page 4-23
IR_WIF_ROLLUPS	on page 4-23

Inference Rule Descriptions

This section provides details about the inference rules listed in the above table. This list of inference rules is in alphabetical order.

IR_ACTIVITY_DATA

The IR_ACTIVITY_DATA inference rule provides all the historical sales activity, beginning with the start date for an item, to the model. The data is loaded on a weekly basis, by week. The view assumes that a week with zero sales is valid for forecasting. A Scenario_ID column is included for use with What If. When What If is invoked from the UI, the override value is entered into Inventory, and Warehouse_Units and On_Order are set to zero.

The field values for Interpretation are:

- 0 = permanent price
- 1 = start of new markdown
- 4 = a price that has not yet been set

IR_BLOCKED_MARKDOWN and IR_BLOCKED_MARKDOWN_C

The IR_BLOCKED_MARKDOWN inference rule is used to indicate the reasons that markdowns are blocked on effective dates. (The exclusion of candidate dates is controlled by IR_MARKDOWN_CALENDAR and the reason for the exclusion is indicated here.) A Scenario_ID column is included for use with What If.

See IR_MARKDOWN_CALENDAR_EX for related information.

IR_BUSINESS_POLICY and IR_BUSINESS_POLICY_C

The IR_BUSINESS_POLICY inference rule provides business constraint information, such as markdown depth and salvage details, that is used by the model run. It looks up most of the values used by the Business Rule Property Manager.

It should produce one row per item to be forecast or optimized in a model run. You will see model configuration errors during a model run if values are incorrect.

For What If, use the Scenario_ID to obtain New_Inventory_Target and New_Salvage_Above_Target from WIF_SCENARIO_TBL.

This inference rule has the following columns:

- Item_ID – the ID of the specified item.
- MinMarkdownInterval – the number of days required between markdowns. This is managed by the NO_TOUCH_AFTER_MKDN business rule.
- MinMarkdownPercentOfFullPrice – the minimum markdown, expressed as a percentage of the original full retail price.
- MaxFirstMarkdownPercentage – the maximum amount for the first markdown, expressed as a percentage of the current permanent price (ticket price). This is managed by the MAX_FIRST_MKDN business rule.
- MaxNumberMarkdowns – The total number of markdowns an item can receive during its life cycle. This is managed by the MAX_MKDN_NO business rule.
- NoMarkdownInPromo – a value, not used by default, that can be used by IR_MARKDOWN_CALENDAR or IR_MARKDOWN_CALENDAR_EX to trigger the elimination of markdown dates that are scheduled during a promotion.

- PromoCeiling – Not used by default. The value can be used by IR_PLANNED_PROMOS to affect Promo Type (interpretation).
- InventoryTarget – the number of items expected to remain unsold by the out-of-stock date (also called out date or exit date). This is managed by the INVENTORY_TARGET business rule as a sell-through percent. The sell-through percent is used to calculate the value for the number of items.
- TargetSellThru – the fraction of inventory that the application should try to sell.
- SalvageValueAboveTarget – the value of an item when the inventory target is not met, expressed as a dollar amount. This is managed by the SALVAGE_ABOVE_TARGET business rule. The dollar amount is used to calculate the salvage value as a percentage of the full retail price.
- SalvageAboveTargetPercent – the salvage value for unsold items above the sell-through target.
- SalvageValueWithTarget – the value of an item when the inventory target is met, expressed as a dollar amount. This is managed by the SALVAGE_WITHIN_TARGET business rule. The dollar amount is used to calculate the salvage value as a percentage of the full retail price. The value is used by IR_PRICE_LADDER.
- DaysAfterLand – the minimum number of days after the first optimization date before the item is eligible for a markdown. This is managed by the NO_TOUCH_AFTER_LAND business rule. It is used by IR_MARKDOWN_CALENDAR to eliminate some potential markdown dates for optimizations.
- NoMarkdownOnEffective – used by IR_MARKDOWN_CALENDAR or IR_MARKDOWN_CALENDAR_EX to eliminate a specific recommended date as an effective markdown date. This value is not a default value.
- MaxMarkdownPercentOfFullPrice - the maximum markdown, expressed as a percentage of the original full retail price. This is used by IR_PRICE_LADDER to trim the list of candidate prices available to the optimization.
- StockoutLevel – used to determine whether or not the inventory target has been met, for purposes of applying salvage targets. The value is expressed in units and is typically set to 0.
- MaxAbsolutePrice – not implemented. Set to 1.
- MarkdownDayOfWeek – can be used by IR_ITEMS_DATES and IR_MARKDOWN_CALENDAR to indicate the day of the week that is the markdown day.
- DaysBeforeOutdate – used by IR_MARKDOWN_CALENDAR or IR_MARKDOWN_CALENDAR_EX to eliminate a specific recommended markdown date that is close to the out date. This value is not a default value.
- MinFirstMarkdownPercentage – the minimum amount for the first markdown, expressed as a percentage of the current permanent price (ticket price). This is managed by the MIN_FIRST_MKDN business rule.
- MinSubseqMarkdownPercentage – the minimum amount for every markdown after the first one, expressed as a percentage of the current permanent price (ticket price). This is managed by the MIN_OTHER_MKDN business rule.
- MaxSubseqMarkdownPercentage – the maximum amount for every markdown after the first one, expressed as a percentage of the current permanent price (ticket price). This is managed by the MAX_OTHER_MKDN business rule.

- TempMarkdownsBlock – Used to indicate whether to consider temporary markdowns when calculating MaxNewMarkdowns and when making decisions based on MinMarkdownInterval.
- PosMarkdownsBlock – Used to indicate whether to consider POS markdowns when calculating MaxNewMarkdowns and when making decisions based on MinMarkdownInterval.
- Scenario_ID – 0 for model run; all other values identify a specific What If scenario.

IR_COLLECTION_INFO

The IR_COLLECTION_INFO inference rule defines information about each pricing group. For the model run, it uses Collection_Pricing to specify the pricing rule. The three pricing rules are:

- Price-together – the pricing recommendations for the items in a group are to the same price points
- Percent-together – the pricing recommendation for the items in a group are to the same percentage off
- Markdown-together – the items in a group are marked down together

This inference rule also supplies the Collection_ID to the Front_End_Collection_ID (pricing group name) mapping for the UI.

This inference rule has the following columns:

- Collection_ID
- Collection_Client_ID
- Collection_Desc
- Parent_Collection_ID
- Land_Dt
- Out_Dt
- Clearance_Ind_Dt
- Price_Ladder_ID
- Clr_Price_Ladder_ID
- Collection_Type – This specifies the business constraints on the markdown recommendations for items in a pricing group, as follows:
  - PriceTogether – all items in a pricing group must be markdown down to the same dollar value.
  - PercentOffTogether – all items in a pricing group must be marked down to the same percentage off the original retail price.
  - MarkdownTogether – all items in a pricing group must be marked down, but the markdown prices have no defined relationship with each other.
- Parent_Collection_Desc
- Parent_Client_ID
- Collection_Pricing
- Is_A_Front_End_Collection
- Front_End_Collection_ID

IR_DISPLAY_PROMOS

The IR_DISPLAY_PROMOS inference rule lists the information about promotions that is displayed in the UI. It is based on IR_PLANNED_PROMOS, with differences as noted below.

This inference rule has the following columns:

- Item_ID – the ID of the item affected by the promotion.
- Price – the promotion price (not the relative price).
- Interpretation – the type of promotion. Interpretation affects the business rules that apply to a given promotion. The business rules affect the legality of the markdowns in the vicinity of the promotion.

  The possible values for interpretation are:

  - Promo_Floor (2) – a floor promotion.
  - Promo_Ceiling (3) – a ceiling promotion.
  - Promo_Unrestricted (9) – a promotion that has no restrictions.
- StartDate – unlike in IR_PLANNED_PROMOS, this start date includes all promotions.
- EndDate – the actual end date (not end_dt + 1).
- Priority – a value used to prioritize all the promotions of a given type in order to eliminate any possible conflicts. The default value is 2.

  The actual precedence rules used to determine the promotion used are:

  1. Floor promos win.
  2. Lowest price.
- Lift – the effect of an external event, such as advertising, on sales when a promotion is in effect. Used in forecasting. A multiplier applied to the demand.
- LiftType – used to define the lift. The possible values for lift are:

  - Base (0) – for base media lifts.
  - Relative (1) – for relative media lifts.
  - POS (2) – for percent-off events that are independent of markdown status.
  - Additional (3) – for percent-off events. Applicable only to items that have had one or more markdowns.
  - No_Markdown (4) – for percent-off events. Applicable only to items that have had no markdowns.
  - First_Markdown (5) – for percent-off events. Applicable only to items that have had one markdown.
  - Multiple_Markdown (6) – for percent off events. Applicable only to items that have had two or more markdowns.

  Base and relative are used for combining media effects. The lift on a given day is computed by multiplying max (Base lifts) and max (Relative lifts). POS, Additional, No_Markdown, First_Markdown, and Multiple_Markdown are all used for point-of-sale promotions. In these promotions, the sales price is calculated by taking a percent off the ticket price. The percent off is specified in the application field as a relative price. So, 35 % off means a relative price of 0.65. The promotional price is triggered only if the specified Lift Type conditions apply.

A POS means that the discount is taken in addition to lowest permanent (list or markdown, but not promotion or clearance) price. Additional means that the discount is taken in addition to the lowest permanent (markdown, but not promotion or clearance) price. The Interpretation for either POS, Additional, No_Markdown, First_Markdown, and Multiple_Markdown promotions should be set to PROMO_UNRESTRICTED.

- Promo_Desc – a description of the promotion.
- Promo_Pct_Off – the actual value (not 1 - Promo_Pct_Off).
- Promo_Type – the type of promotion.
- Promo_Number – the number identifying the promotion.
- Attributes 1-5 – variable attributes.
- Week_End_Date – the date for the last day of the week.

IR_ELIGIBLE

The IR_ELIGIBLE inference rule is used to provide a list of the eligible items and eligible pricing groups to the model run. Eligibility is defined and customized using the load statements.

IR_FE_WAREHOUSE

The IR_FE_WAREHOUSE inference rule references the IR_WAREHOUSE view. It lists the warehouse on-hand and on-order units for an item.

IR_FORCED_MARKDOWNS

The IR_FORCED_MARKDOWNS inference rule defines the markdown level an item is required to have. If the item has not reached the defined markdown level by the scheduled time, then a markdown will be forced even if it is not desirable, or the opportunity cost will be zero.

IR_FORECAST_METRICS

The IR_FORECAST_METRICS inference rule contains a list of forecasted metrics.

This inference rule has the following columns:

- Ending_Inventory_Units
- EOL_Cum_Unit_Sales
- EOL_Cum_Dollars_Sales
- Weekly_Projected_Unit_Sales
- Weekly_Projected_Dollar_Sales
- Weekly_Projected_Sales_Price
- Projected_Out_of_Stock
- Rec_Rtl_Min
- Forecast_ID

IR_FORECAST_METRICS_POSTRUN

For What If, the Scenario_ID is specified in internal queries. Used for updating ITEM_DATA in the POSTRUN step.

IR_FRONT_END_IDS

The IR_FRONT_END_IDS inference rule provides the Store_ID, Merchandise_ID, Ladder_ID, and Current_Ladder_ID associated with an item to the ITEM_DATA table.

IR_HISTORIC_METRICS

The IR_HISTORIC_METRICS inference rule lists the following historic metrics:

- Cumulative quantity sold
- Cumulative sales dollars
- Current on order dollars
- Inventory cost
- Current units on order
- Start sell date
- Week-minus-1 units on hand
- Week-minus-2 units on hand
- Week-minus-3 units on hand
- Unit sales through week
- Unit sales week-minus-1
- Unit sales week-minus-2
- Unit sales week -minus-3
- Dollar sales through week
- Dollar sales week-minus-1
- Dollar sales week-minus-2
- Dollar sales week-minus-3

IR_ITEM_BASE_DEMAND

The IR_ITEM_BASE_DEMAND inference rule is used by Analytical Services to apply (or override) a demand strategy to historical sales.

This inference rule has the following columns:

- Item_ID – identifies the item.
- Base_Demand – the external base demand value, which must be positive, or it will be ignored.
- Base_Demand_Usage – must have a value of either Override or Floor. If set to Override, it overrides the internal base demand calculated by the Calculation Engine and uses the external base demand calculation. If set to Floor, the internal value is used.

IR_ITEM_COLLECTION

The IR_ITEM_COLLECTION inference rule defines how items are grouped into pricing groups.

This inference rule has the following columns:

- Item_ID
- Merchandise_ID

- Location_ID
- Collection_Client_ID
- Collection_Desc

This inference rule can be configured with a custom list of excluded/included items. It works in combination with IR_ITEM_COLLECTION_OPTION.

IR_ITEM_COLLECTION_OPTION

The IR_ITEM_COLLECTION_OPTION inference rule includes a flag by default set to *N*, which indicates that the pricing groups are managed at the level of optimization. The *Y* flag is used to indicate pricing group management at the Chain level (optimization is still at the item level).

IR_ITEM_DAILY_WEIGHTS_OVERRIDE

The IR_ITEM_DAILY_WEIGHTS_OVERRIDE inference rule provides daily weights that override the default daily weights for an item. The Calc Engine uses daily weights in combination with the weekly forecasted sales units to determine daily forecasts.

IR_ITEM_DAILY_WEIGHTS_OVERRIDE has 8 columns: ITEM_ID, SUNDAY_WT, MONDAY_WT, TUESDAY_WT, WEDNESDAY_WT, THURSDAY_WT, FRIDAY_WT, and SATURDAY_WT to accommodate a weight for each weekday and for each item whose daily weights need to be overridden. By default, this view has no records, so no daily weights are overridden.

Table 4–4 Default Daily Weight Values

Day of Week	Default Daily Weight Value
Sunday	0.2
Monday	0.1
Tuesday	0.1
Wednesday	0.1
Thursday	0.1
Friday	0.2
Saturday	0.2

The view is not required to have any record unless daily weights need to be overridden for some items. For items that do not have record in the view, the CE will apply the default daily weights.

To override the daily weights of an item, this view must contain exactly one record for the ITEM_ID. The CE checks the validity of the daily weight record in two steps:

First, the weight for each weekday must be a non-negative double value, and weights for the same item must add up to 1. If any weekday has a negative value or a NULL value, or the values do not add up to 1, the CE will throw a "badDailyWeightsOverride" error and will not generate a forecast or markdown recommendation.

Second, the daily weights are compared with the historic data. If a particular weekday has a daily weight value of 0, but the historic daily sales on the same weekdays do not have a value of 0, the CE will throw a "badSeasonality" error. However, the CE does allow the daily weight and the daily sales on the same weekday to both have a value of 0 at the same time; in this case, it will forecast zero sales on the same weekday.

Here is the default ir_item_daily_weights_override from ir.sql:

```
CREATE VIEW IR_ITEM_DAILY_WEIGHTS_OVERRIDE
(
    ITEM_ID,
    SUNDAY_WT,
    MONDAY_WT,
    TUESDAY_WT,
    WEDNESDAY_WT,
    THURSDAY_WT,
    FRIDAY_WT,
    SATURDAY_WT
)
AS
  SELECT
    i.item_id,
    0.2 as SUNDAY_WT,
    0.1 as MONDAY_WT,
    0.1 as TUESDAY_WT,
    0.1 as WEDNESDAY_WT,
    0.1 as THURSDAY_WT,
    0.2 as FRIDAY_WT,
    0.2 as SATURDAY_WT
  FROM
    items_tbl i
 WHERE i.end_dt IS NULL
 AND 1 = 0
%{YA_TD}%
```

Here is an example of a customized version of the view in which all the items share the same daily weights that are different from the default ones.

```
CREATE VIEW IR_ITEM_DAILY_WEIGHTS_OVERRIDE
(
    ITEM_ID,
    SUNDAY_WT,
    MONDAY_WT,
    TUESDAY_WT,
    WEDNESDAY_WT,
    THURSDAY_WT,
    FRIDAY_WT,
    SATURDAY_WT
)
AS
  SELECT
    i.item_id,
    0.0 as SUNDAY_WT,
    0.1 as MONDAY_WT,
    0.1 as TUESDAY_WT,
    0.2 as WEDNESDAY_WT,
    0.1 as THURSDAY_WT,
    0.2 as FRIDAY_WT,
    0.3 as SATURDAY_WT
  FROM
    items_tbl i
 WHERE i.end_dt IS NULL;
```

IR_ITEM_DATES and IR_ITEM_DATES_C

The IR_ITEM_DATES inference rule defines a set of intervals, beginning with the start date and ending with the out date. StartDate is defined as Sunday by default.

For What If, use the Scenario_ID to obtain New_Out_Dt from WIF_SCENARIO_TBL.

This view assumes an updated ITEMS_BRM_RULES table that contains current outdate values.

Note that days of the week must be aligned correctly or errors will result.

This inference rule has the following columns:

- ItemID – identifies the item the dates apply to.
- StartDate – the first date that an item is considered to be available for sale. It is not the date on which the item arrives in the store or the date of the first sale. It can be calculated based on sales or it can be supplied directly from the retailer through a data feed.
- StartSimulationDate – the date on which the simulation starts, which is defined by default by adding one day to the last day of historical activity. The last day of history is always a Saturday, which is the last day that the application has the sales data from the retailer.
- EffectiveDate – the date on which a new markdown recommendation from the run can be applied to the item. This date is generally the one on which the new markdown is possible, given the production cycle. If the model run makes a markdown recommendation for this day, then it will be available for approval in the application UI. It is usually x days after the last day of history. Some retailers may have varying effective dates for different departments.
- OutDate – the date on which all items are sold or the target inventory value is met. The value for OutDate in IR_ITEM_DATE and IR_ITEM_DATE_C must be aligned. The use of ITEMS_MODELRUN_TBL for outdates is not appropriate.
- DB_Last_Actual_Date – the last day of historical activity.
- Scenario_ID – 0 for model run; all other values identify a specific What If scenario.

IR_ITEM_IDS and IR_ITEM_IDS_C

The IR_ITEM_IDS inference rule provides a set of IDs that are associated with an item.

This inference rule has the following columns:

- Item_ID – identifies the item.
- Collection_ID – used only with IR_ITEMS_IDS_C.
- Merchandise_ID – used in association with the Location_ID to identify an item.
- Location_ID – used in association with the Merchandise_ID to identify an item.
- Price_Ladder_ID – identifies the price ladder associated with an item.
- Seasonality_ID – uses the seasonality attribute value, which identifies the seasonality curve for the item, and that is defined in IR_SEASONALITY_ ATTRIBUTE.

IR_ITEM_INFO and IR_ITEM_INFO_C

The IR_ITEM_INFO inference rule shows basic information about an item, including price and date. This view references IR_ITEM_DATES and IR_ITEM_PRICES. For What If, Scenario_ID is specified in internal queries.

IR_ITEM_PARAMETERS

The IR_ITEM_PARAMETERS inference rule defines the analytical parameters used in a forecast and are provided by Analytical Services. This view includes the following

columns: Item_ID, Gamma, CriticalInventory, ZeroInventoryEffect, Demand_Uncertainty, Model, Demand_Strategy, Demand_Intervals, MaxNewMarkdowns, Alpha, Beta, PriceEffect, InSeasonDistribution, InSeasonParameter, UseInternalPrior, Lambda, and InternalPriorBias.

IR_ITEM_PRICES and IR_ITEM_PRICES_C

The IR_ITEM_PRICES inference rule provides the basic set of prices for each item. One row must exist for each item (an eligible item) run through the model. The Perm_Ticket_Price only reflects markdowns from model runs, not from What If calculations. The value should be the ticket price as of the effective date.

The view contains "scenario_id=0" to ensure that What If data is not accessed.

Note that the "item does not exist" error is primarily caused by a failure in this query.

This inference rule has the following columns:

- Item_ID
- Full_Price
- Ticket_Price
- Perm_Ticket_Price
- Current_Inv_Price
- Avg_Cost

IR_LOC_OPT_LEVEL

This inference rule provides the location optimization level, as defined in the Cross Products Information Standard Interface.

IR_LOCATION_HIERARCHY

The IR_LOCATION_HIERARCHY defines an item's location hierarchy.

IR_MARKDOWN_CALENDAR

The IR_MARKDOWN_CALENDAR inference rule defines the markdown calendar for an item. These dates are used during an optimization. The view can be used, for example, to trim the calendar so that there are no markdowns during the last weeks. The item dates view and the markdown calendars view share common logic.

See IR_MARKDOWN_CALENDAR_EX for related information. This view is necessary when a popup message explaining the exclusion is needed.

This rule produces zero or more rows representing recommended markdown dates. If the eligible effective date is not available, or if this view returns zero rows, then markdowns are not recommended. Markdowns can only be recommended if available effective dates are provided by this view. The dates are based on the weekly calendar provided by the retailer. It uses IR_BUSINESS_POLICY and IR_PLANNED_PROMOS to apply restrictions to the set of recommended dates. Additional restrictions may also be applied, based on IR_MARKDOWN_CALENDAR_EX.

For What If, use the scenario_ID to obtain the New_Blackout_End from WIF_SCENARIO_TBL.

This inference rule has the following columns:

- ItemID – the ID of the item being marked down.

- CalendarDate – the date of the candidate markdown. This should be between the effective date and the out date.
- Scenario_ID – 0 for model run; all other values identify a specific What If scenario.

IR_MARKDOWN_CALENDAR_EX and IR_MARKDOWN_CALENDAR_EX_C

The IR_MARKDOWN_CALENDAR_EX inference rule defines the dates that are excluded from the standard markdown calendar. It excludes dates from IR_MARKDOWN_CALENDAR and provides reason codes for the exclusion to IR_BLOCKED_MARKDOWN. The view uses resource IDs to describe the reason for the exclusion, so use only properly defined resources. A Scenario_ID column is included for use with What If.

IR_MERCH_OPT_LEVEL

This inference rule provides the merchandise optimization level, as defined in the Cross Products Information Standard Interface.

IR_MERCHANDISE_HIERARCHY

The IR_MERCHANDISE_HIERARCHY inference rule defines an item's merchandise hierarchy.

IR_METRICS

The IR_METRICS inference rules lists the following metrics:

- Unit cost
- MTD net sales units
- MTD net sales amount
- STD net sales units
- STD net sales amount

IR_MISSING_WEEKS

The IR_MISSING_WEEKS inference rule defines the weeks in an item's history that are missing activities. An item should have, at a minimum, history from its start date to the last day of history. A Scenario_ID column is included for use with What If.

IR_MODEL_START

The IR_MODEL_START inference rule is used when the IR_MODEL_START_OPTION is defined as custom. It defines the model start date for items and produces one row per item.

This inference rule has the following columns:

- Item_ID – identifies the item.
- Model_Start_Dt – the first date that the item is available for sale.

IR_MODEL_START_OPTION

The IR_MODEL_START_OPTION inference rule is used to configure the Option and Threshold (when necessary) settings to determine the model start date (the first possible sale date for an item) used for optimizations. This inference rule produces a single row containing a global setting. This view should be used for configuring Option and Threshold for setting Model_Start_Dt in ITEMS_TBL. (Model_Start_Dt is

always represented as the first day of the week preceding the actual computed date. It is loaded using LoadModelStartDate, which is part of plfrontendload.sh (FELOAD).)

This inference rule has the following columns:

- Model_Start_Option – the value must be one of the following:
  - inventoryRatio – (inventory/cumulative_sales_to_date + inventory + on_order) above a defined Threshold.
  - storeRatio – (stores_with_inventory/stores_in_region) above a defined Threshold.
  - plannedStart – the default. No threshold value needed.
  - custom – derives the value from IR_MODEL_START. No threshold value needed.
  - sellThrough – used when Model Start Sell Through Pct is used to determine the model start date. When this option is selected, then a value (Y or N) must be provided for Recalc, Use_StoreOH_Inv, Use_StoreOO_Inv, Use_DCOH_Inv, and Use_DCOO_Inv.

    If the value of either Use_StoreOH_Inv, Use_StoreOO_Inv, Use_DCOH_Inv, and Use_DCOO_Inv is set to Y (the default), then the specified value for that parameter is used in the calculation of the Model Start Date. The total inventory is calculated by summing the store and distribution center inventories. Most retailers use the following combinations:

    * Store On Hand + Store On Order
    * Store On Hand + Store On Order + DC On Hand
    * Store On Hand + Store On Order + DC On Hand + DC On Order

    Note that if the Sell Through option is used, then the three business rules, MSD_FORCED_START_DT, MSD_SELLTHROUGH_PCT, and MSD_MAX_DELAY_WK, must be configured.
- Threshold – the numeric value of the threshold, for inventoryRatio and storeRatio only. This value must be between 0 and 1.
- Recalc – used to indicate that a recalculation will be used. The default value is Y.
- Use_StoreOH_Inv – the value for Store On Hand Inventory is used in the calculation of Model Start Date. The default value is Y.
- Use_StoreOO_Inv – the value for Store On Order Inventory is used in the calculation of Model Start Date. The default value is Y.
- Use_DCOH_Inv – the value for DC On Hand Inventory is used in the calculation of Model Start Date. The default value is Y.
- Use_DCOO_Inv – the value for DC On Order Inventory is used in the calculation of Model Start Date. The default value is Y.

IR_MODEL_VALUES

The IR_MODEL_VALUES are provided by Analytical Services.

IR_O_USER_DATES and IR_O_USER_DATES_C

The IR_O_USER_DATES inference rule defines six date values per item per run ID, based on retailer requirements, for the ITEM_DATA table during the model run. This inference rule does have access to forecast and markdown information.

This inference rule has the following columns:

- Item ID – identifies the item
- User date columns as appropriate

IR_O_USER_FLOATS and IR_O_USER_FLOATS_C

The IR_O_USER_FLOATS inference rule defines twelve numeric values per item per run ID, based on retailer requirements, for the ITEM_DATA table during the model run. This inference rule does have access to forecast and markdown information.

This inference rule has the following columns:

- Item ID – identifies the item
- User float columns as appropriate

IR_O_USER_TEXTS and IR_O_USER_TEXTS_C

The IR_O_USER_TEXTS inference rule defines four text values per item per run ID, based on retailer requirements, for the ITEM_DATA table during the model run. This inference rule does have access to forecast and markdown information.

This inference rule has the following columns:

- Item_ID – identifies the item
- User text columns as appropriate

IR_P4P_ITEMS_CONFIG

This inference rule provides a single point of configuration for markdown information. The initial definition of the view is a pass through to the P4P_ITEMS view.

The P4P_DISPLAY_ITEMS view has been modified to be a join between P4P_ITEMS and IR_P4P_ITEMS_CONFIG. The view now returns TAKEN_PRICE instead of PROPOSED_PRICE. PL_MARKDOWN_SENDBACK, RDM load, sample reports, all XML configurations, P4P_DISPLAY_ITEMS, and P4P_MAINTAIN_ITEMS have been changed to use or return TAKEN_PRICE.

The Proposed_Price column (as well as the Int_Proposed_Price column in p4p-column-list.xml) should be used to reflect the value of the price ladder. The new column, Int_Taken_Price in p4p-column-list.xml, maps to the Taken_Price column in P4P_DISPLAY_ITEMS. This column has been added to all grids that contain Proposed_Price.

The IR_P4P_ITEMS_CONFIG has the following columns:

- Item_ID
- Submittal_Worksheet_ID
- Markdown_Flag
- Recommended_Retail_Price
- Taken_Price

The initial definition of the view is a pass-through to the p4p_items view.

The markdown_flag column accepts the following values:

- 0 – markdown not taken
- 1 – markdown taken to item recommended price
- 2 – markdown taken to pricing group recommended price

- 3 – markdown taken to user modified price
- 4 – markdown taken due to optimization to budget
- 5 – markdown taken due to What If

IR_P4P_MARKDOWN_ACTIVITIES

The IR_P4P_MARKDOWN_ACTIVITIES inference rule is used to return markdown activities to What If that match the forecasts in P4P_FORECAST_DATA. These forecasts reflect whether pricing groups or items were used in the model run, so this view keeps What If consistent.

IR_PAST_TICKET_PRICES

The IR_PAST_TICKET_PRICES inference rule provides a ticket price history to the model. This information is used to determine the number of markdowns that have already occurred, the date of the last markdown, and the starting ticket price for the forecast. A Scenario_ID column is included for use with What If.

The field values for interpretation are:

- 0 = permanent price
- 1 = start of new markdown
- 4 = unknown price

IR_PENDING_MARKDOWNS

The IR_PENDING_MARKDOWNS inference rule defines markdowns that have already been accepted but are still in the forecast range and so should be taken into account by the forecast. Markdowns from two different sources are included:

- historic markdowns, taken during previous weeks, that are not yet in the sales history
- markdowns proposed by What If

This view handles markups, markdowns, and any pending price changes. The view returns the relative price as taken from the full price (or the current ticket price if the interpretation is 3). It is a multiplier applied to the original retail (full) price for the PERM and TEMP interpretations, and to the current ticket price for the POS interpretation.

This may require customization so that Start_Dt occurs on the correct date.

Temporary markdowns are flagged to distinguish them from POS markdowns.

For What If, use the scenario_ID to obtain Item_Markdowns from WIF_ITEM_MARKDOWN_TBL.

This inference rule has the following columns:

- ItemID – identifies the item associated with the markdown.
- StartDate – the effective date of the markdown.
- Price – the relative price, a multiplier that is applied to the original retail price (full price) for interpretations 1 and 2 and to the current ticket price for interpretation 3.
- Interpretation – Permanent markdown = 1. Temporary markdown = 2. POS markdown = 3.
- Scenario_ID – 0 for model run; all other values identify a specific What If scenario.

IR_PLANNED_PROMOS

The IR_PLANNED_PROMOS inference rule defines the characteristics of all future planned temporary markdowns and the associated expected lift for each item. In the forecasted range, this is used to determine the current selling price and to implement floor and ceiling restrictions on markdowns. Promotions with lifts are determined based on a historical analysis of an item's demand.

A Scenario_ID column is included for use with What If. The value returned by this view does not vary by scenario ID. The view depends on IR_ITEM_DATES and IR_PENDING_MARKDOWNS (using IR_ITEM_PRICES). These two views do contain override logic. IR_PLANNED_PROMOS does not depend on fields that vary with What If. The Scenario_ID column is included in this view to provide robustness during customization.

This inference rule has the following columns:

- Item_ID – the ID of the item affected by the promotion.
- Price – the relative price. Price affects demand according to the price effect function.
- Interpretation – the type of promotion. Interpretation affects the business rules that apply to a given promotion. The business rules affect the legality of the markdowns in the vicinity of the promotion.

  The possible values for interpretation are:

  - Promo_Floor (2) – a floor promotion.
  - Promo_Ceiling (3) – a ceiling promotion.
  - Promo_Unrestricted (9) – a promotion that has no restrictions.
- StartDate – the date on which the promotion will begin.
- EndDate – the date on which the promotion will end.
- Priority – a value used to prioritize all the promotions of a given type in order to eliminate any possible conflicts. The default value is 2.

  The actual precedence rules used to determine the promotion used are:

  1. Floor promos win
  2. Lowest price
- Lift – the effect of an external event, such as advertising, on sales when a promotion is in effect. Used in forecasting. A multiplier applied to the demand.
- LiftType – used to define the lift. The possible values for lift are:
  - Base (0) – for base media lifts.
  - Relative (1) – for relative media lifts.
  - POS (2) – for percent-off events that are independent of markdown status.
  - Additional (3) – for percent-off events. Applicable only to items that have had one or more markdowns.
  - No_Markdown (4) – for percent-off events. Applicable only to items that have had no markdowns.
  - First_Markdown (5) – for percent-off events. Applicable only to items that have had one markdown.

- Multiple_Markdown (6) – for percent off events. Applicable only to items that have had two or more markdowns.

Base and relative are used for combining media effects. The lift on a given day is computed by multiplying max (Base lifts) and max (Relative lifts). POS, Additional, No_Markdown, First_Markdown, and Multiple_Markdown are all used for point-of-sale promotions. In these promotions, the sales price is calculated by taking a percent off the ticket price. The percent off is specified in the application field as a relative price. So, 35 % off means a relative price of 0.65. The promotional price is triggered only if the specified Lift Type conditions apply.

A POS means that the discount is taken in addition to lowest permanent (list or markdown, but not promotion or clearance) price. Additional means that the discount is taken in addition to the lowest permanent (markdown, but not promotion or clearance) price. The Interpretation for either POS, Additional, No_Markdown, First_Markdown, and Multiple_Markdown promotions should be set to PROMO_UNRESTRICTED.

- Scenario_ID - 0 for model run; all other values identify a specific What If scenario.

IR_PRICE_LADDER and IR_PRICE_LADDER_C

The IR_PRICE_LADDER inference rule sets the available prices that the model can use for optimization. The prices in the price ladder are defined relative to the original full retail price. This can be customized to trim the available prices based on specific business constraints.

This inference rule defines the base candidate prices that are used on any available markdown day. The optimizer uses this to determine an optimal sequence of markdowns. So assumptions based on the current date or the current price should be carefully evaluated.

The prices supplied by IR_PRICE_LADDER_C are assumed to be consistent with the pricing group pricing rule. Otherwise, they will not be considered as markdown candidates. For example, in a "price together" pricing group, only the dollar amounts (calculated using relative price * full price) common to each item's price ladder will be considered. If there are no dollar values in common, no markdown is possible.

A Scenario_ID column is included for use with What If.

This inference rule has the following columns:

- Item_ID – identifies the item.
- Price – the price relative to the full price for the item.
- Interpretation – Permanent markdown = 1.
- Scenario_ID – 0 for model run; all other values identify a specific What If scenario.

IR_PRIOR_DISTRIBUTION

The IR_PRIOR_DISTRIBUTION inference rule lists Analytical Services values.

IR_PROCESS_NULL_OUTDT

This inference rule is used for backward compatibility with versions of the application prior to 4.0.

IR_PROJ_MKDNS

The IR_PROJ_MKDNS inference rule. For What If, Scenario_ID is specified in internal queries.

IR_ROLLUPS

The IR_ROLLUPS inference rule references the ITEM_DATA table directly and calculates rollups based on the data in this table. It lists the following metrics:

- Cumulative average price
- Cumulative sell-through percent
- Cumulative percent off
- Average price for the current week
- Cumulative gross profit percent
- Sell-through percent for the current week
- Sell-through percent week-minus-1
- Sell-through percent week-minus-2
- Sell-through percent week-minus-3
- EOL gross margin dollars
- EOL gross margin percent
- Weeks of supply

IR_SEASON_METRICS

The IR_SEASON_METRICS inference rule lists the following metrics:

- MTD average price
- Unit sales season-minus-1
- STD gross margin percent

IR_SEASONALITY_ATTRIBUTE

The IR_SEASONALITY_ATTRIBUTE inference rule defines the item attribute value that is used to look up seasonalities.

This inference rule has the following columns:

- Item_ID – identifies the item.
- Item_Attribute – the Analytical Services value assigned to the item.

IR_STATE_TRANS_CONFIG_OVERRIDE

The IR_STATE_TRANS_CONFIG_OVERRIDE inference rule is used for risk rating and is provided by Analytical Services.

IR_STATE_TRANS_PREV_RUN

The IR_STATE_TRANS_PREV_RUN inference rule is used for risk rating and is provided by Analytical Services.

IR_USER_DATES and IR_USER_DATES_C

The IR_USER_DATES inference rule defines six date values per item, based on retailer requirements, for the ITEM_DATA table during the model run. This inference rule does not have access to forecast information.

This inference rule has the following columns:

- Item_ID – identifies the item

- User date columns as appropriate

IR_USER_FLOATS and IR_USER_FLOATS_C

The IR_USER_FLOATS inference rule defines twelve numeric values per item, based on retailer requirements, for the ITEM_DATA table during the model run. This inference rule does not have access to forecast information.

This inference rule has the following columns:

- Item_ID – identifies the item
- User float columns as appropriate

IR_USER_TEXTS and IR_USER_TEXTS_C

The IR_USER_TEXTS inference rule defines four text values per item, based on retailer requirements, for the ITEM_DATA table during the model run. This inference rule does not have access to forecast information.

This inference rule has the following columns:

- Item_ID – identifies the item
- User text columns as appropriate

IR_WAREHOUSE

The IR_WAREHOUSE inference rule provides Warehouse_On_Hand and Warehouse_On_Order to the ITEM_DATA table. If a retailer does not want to include warehouse on order units in the forecast, the Warehouse_On_Order units should be set to zero.

WIF_FORECAST_DATA

The WIF_FORECAST_DATA inference rule is dependent on other, configurable inference rules and is included in ir.sql because it cannot be created unless several other inference rules have already been created. It must not be configured or modified.

IR_WIF_PROJ_OH_UNITS_EFF_DT

This view calculates projected on hand units as of the effective date for the item. It is used in What If scenarios. This metric is not calculated for pricing groups. When you use this IR, always add a WHERE clause that assigns values for item_id, submittal_worksheet_id, and forecast_id.

This inference rule has the following columns:

- Item_ID
- Forecast_ID
- Submittal_Worksheet_ID
- Collection_Forecast_ID

IR_WIF_ROLLUPS

Calculates rollup values for an item based on What If scenarios using WIF_KPI_TBL and ITEM_DATA. Contains the following metrics: Proj_Std_EOL, GM_Amount, Proj_Std_EOL_GM_Perc Proj_Std_EOL_GM_Amount_C, and Proj_Std_EOL_GM_Perc_C. This IR is similar to IR_ROLLUPS.

This inference rule has the following columns:

- Item_ID

- Forecast_ID
- Submittal_Worksheet_ID
- Proj_Oh_Units_Eff_Dt

IR_WORKSHEET_IDS

The IR_WORKSHEET_IDS inference rule populates all values up to the level at which worksheets are defined and specifies how the worksheets are mapped to the back end. It generates submittal_worksheet_IDs for the application. Worksheets must be defined for all *N* levels in the combined merchandise hierarchy/location hierarchy, where *N* is a value between 1 and 4.

Defining a Worksheet.

Worksheets are defined at a specific level in the merchandise and location hierarchy and can be defined up to four levels.

To define a worksheet:

1. In ASH_CP_TBL, specify the merchandise and location hierarchy levels for the intersect name WORKSHEET. For example:

INTERSECT_NAME	MERCHANDISE_LEVEL	LOCATION_LEVEL
OPTIMIZATION	COLOR	STORE_CLSTR
WORKSHEET	DEPARTMENT	CHAIN
CLUSTER	CHAIN	CHAIN
DEFAULTLEVEL	CHAIN	CHAIN
SALES	COLOR	STORE

2. Verify that IR_WORKSHEET_IDS has the correct settings.
3. Set the attribute hierarchy-levels-above-worksheet in the <client> tag in p4pgui-config.xml. The attribute level must equal the worksheet level. If the worksheet level = 3, then hierarchy-levels-above-worksheet = 3.
4. In p4p-custom-columns.xml, set the INT_WKSHT_HIERARCHY element to the appropriate hierarchy that corresponds to the worksheet level.

5

What If

The What-If functionality is used to experiment with different optimization strategies.

This chapter contains the following sections:

Introduction

The Markdown Optimization What If functionality allows users to select a group of items, make experimental changes to certain settings, and then perform a re-optimization in order to model the effects of the setting changes on the application markdown recommendations and forecasts for the selected items. If the results are satisfactory, the changes can be applied permanently.

The changes that can be made within the What If functionality are changes that are also available within the application using the Item Maintenance and Take Markdowns Advanced screens. What If simply allows users to experiment with changes on a small group of items and simulate the results.

Any changes taken permanently must be consistent with permissions settings in the Business Rule Property Manager and User Management.

Markdown changes taken within What if are in addition to pending markdown changes.

The information from a given What If recalculation is available only for the length of the specific What If session.

KPIs are not calculated by What If.

What If functionality is implemented within the application through an API call to the Calculation Engine using the RMI Server.

Setting changes are implemented just below the inference rule level so that the inference rules can pick up the changed values. Relevant inference rules have been modified to enable the What If to function with existing inference rules.

The What If end-user functionality is accessed from the application Worksheet and is described in detail in the *Oracle Retail Markdown Optimization User Guide*.

This chapter provides details about the configuration of What If.

Configuring the RMI Server

The RMI server, which is part of the Calculation Engine and which facilitates remote Java method calls, provides What If with access to the Calculation Engine. Each instance of p4pgui is associated with a single RMI server.

Starting the RMI Server and Registry

The enginectl.sh script is used to start and stop the RMI server from the command line. It calls runInteractiveCE.sh with the -p flag set. So, in production, protected mode is the default. This option can only be turned off by modifying the CONFIGURATION section of enginectl.sh.

To start, stop, kill, restart, get the status for, or get help about the interactive engine (RMI server), use the following commands:

`enginectl.sh <ConfigRoot> start`
This starts the engine and the failover process.

`enginectl.sh <ConfigRoot> stop`
This stops the engine and the failover process and provides an error message on failure.

`enginectl.sh <ConfigRoot> kill`
This stops the engine and the failover process and provides an error message on failure.

`enginectl.sh <ConfigRoot> restart`
This stops and then restarts both the engine and the failover process. It provides an error message if restart fails. It does not provide an error message if stop fails and restart succeeds.

`enginectl.sh <ConfigRoot> status`
This message indicates whether or not the engine is running and if the protected mode flag is set.

`enginectl.sh <ConfigRoot> help`
This prints a usage message.

Other versions of the help command include enginectl.sh help and enginectl.sh.

These commands are located in modules/tools/bin.

Port and Host Configuration

The port used by the RMI server must be configured as delphi.rmi.port in:

- config/Engine/delphi.properties
- config/suite/suite.properties

The value assigned to delphi.rmi.port must be the same in both files.

In addition, a value must be assigned to delphi.rmi.host in config/suite/suite.properties.

For example:

delphi.rmi.host=orr.grossprofit.com

delphi.rmi.port=7062

What If and Pricing Groups

The Markdown Optimization model run can be configured to optimize items as both a member of a pricing group and as an individual item. What If recalculations can be configured to optimize an item either as a member of a pricing group or as a an individual item.

You configure the What If setting in config/Price/config.properties.

The default setting specifies that a What if recalculation occurs at the item level:

pricefe.whatif.itemDominant=true

To specify that a What If recalculation occurs at the pricing group level:

pricefe.whatif.itemDominant=false

Markdown Optimization has three configuration points for item vs. pricing group optimization and these should be configured consistently:

- P4P_FORECAST_DATA table, which is populated using load_Statements.sql
- IR_P4P_MARKDOWN_ACTIVITIES, which is used by What If to access model run recommendations
- pricefe.systemwide.ItemDominant (default = true) in config.properties

What If displays the forecasts and recommendations from the model run by querying P4P_FORECAST_DATA and markdowns taken from IR_P4P_MARKDOWN_ ACTIVITIES. So, the What If display information reflects the configuration of the three configuration points. However, the What if recalculation reflects only the configuration of pricefe.whatif.itemDominant setting.

What If Size Limitations

What If recalculations perform best within certain size limitations. You can configure What If to define the number of items that are permitted in a given recalculation.

The following parameters are configured in config/p4pgui/config.properties:

- p4pgui.what-if.max.size - should not be set to greater than 1,000. The recalculation will not be initiated if the number of items exceeds this value. Use this parameter to manage how users can configure What If so that performance is not degraded.
- p4pgui.what-if.warn.size - best if set to 100. Setting this to a higher value can impact performance. The recalculation will be initiated if the number of items exceeds this value; however, the user will receive a warning.
- p4pgui.what-if.pricing-group-item.weight - use this value as a variable when recalculating items in a pricing group. The value reflects the relative cost of optimizing individual items as compared to optimizing items in pricing groups.

Configuring p4pgui config.properties

The following settings must be configured to determine pricing group membership in order to determine which items to re-optimize. For more information on What If and pricing groups, see "What If and Pricing Groups" on page 5-3.

Table 5–1 What If Settings in config.properties

Setting	Definition	Value
pricefe.what-if.itemDominant (See also pricefe.systemwide.itemDominant, described in the chapter config.properties)	Determines whether What If re-optimizes an item as an item or as a pricing group member	True (default) - items re-optimized as items False - items re-optimized as pricing group members
p4pgui.max.what-if.rows	Obsolete	NA
p4pgui.what-if.max.size	Defines the hard limit on the number of items	Default = 1000
p4pgui.what-if.warn.size	Defines the soft limit on the number of items	Default = 100
p4pgui.what-if.pricing-group-item.weight	Defines the weight for each additional item in a pricing group being re-optimized	Usually a decimal value less than 1.0. Default value = 0.7.

Front End Configuration for What If

The What If display in the application UI provides functionality to allow the user to view the settings for the scenario variables and to enter override values that are used in the What If recalculation. The What If display, including display text, input formats, and output formats, is configurable. For more information about using the What If user interface, see the *Oracle Retail Markdown Optimization User's Guide*.

Configuring the Scenario Settings Display

The Scenario Settings display can be configured in two ways. Each row in the display grid can be visible or hidden. The Override Value in each row is either editable or not editable. These settings are configured in the config/Price/grids/p4p-what-if-scenario-variables.XML file. Each configuration is provided by a custom property for each row-group. Each row-group represents one row in the grid.

For example:

```
<row-group>    <
key>SALVAGE_VALUE</key>
    <rowgroup-properties/>
    <custom-property name="hidden" value="false" custom-type="application"/>
    <custom-property name="editable" value="true" custom-type="application"/>
</row-group>
```

Each column header has a resource key in config/Price/resources/p4pguiResources.properties. Each row key is associated with the label for the Scenario Variable column. Appropriate formats for input and output strings are also configured here.

For example:

```
#column heading
p4pgui.whatIf.scenario.variable.heading = Scenario<br>Variable
```

```
#row resources for Current Inventory scenario variable
p4pgui.whatIf.scenario.name.currentInventory = Current Inventory
p4pgui.whatIf.scenario.format.currentInventory = #0.00%
p4pgui.whatIf.scenario.input.format.currentInventory = #0.00
p4pgui.whatIf.scenario.range.currentInventory = {0} - {1}
```

It is possible to un-comment the appropriate format patterns so that the user does not have to enter the % symbol. For example:

```
#p4pgui.whatIf.scenario.format.currentInventory = #0.00%
p4pgui.whatIf.scenario.input.format.currentInventory = #0.00
```

However, these formats must be synchronized with the default formats found in CommonMessages.properties.

Configuring the What If Display/Metrics

Pre-defined metrics, which are listed in "What If Metric Calculations" on page 5-10, are configured in config/Price/grids/p4pgui-config.xml.

For example:

```
<what-if-view-row
    display-name="p4pgui.whatIfRow.markdownDollars.label"
    description="p4pgui.whatIfRow.markdownDollars.description"
    use-as="markdownDollars" type="money"
    format="p4pgui.whatIfRow.markdownDollars.format"/>
```

What If and the Database

The output from a What If recalculation is stored in the database, as follows:

- the recommended markdowns (MARKDOWN_ACTIVITIES), WIF_ITEM_MARKDOWN_TBL
- the override values - WIF_SCENARIO_TBL

ITEM_DATA is not written to by What If, so the model run values are maintained there.

KPIs are not calculated by What If.

The cleanup of this output occurs automatically on a per-session basis. After cleanup, WIF_SCENARIO_TBL is truncated and is re-seeded with Scenario_ID=0. For example:

```
insert into WIF_SCENARIO_TBL
values (0, null, null, null, null, null)
```

The records in the RTM tables for What If contain the appropriate Scenario_ID.

Database Tables

Here are some of the tables associated with What If:

WIF_SCENARIO_TBL - contains the scenario override values for a given Scenario_ID.

Table 5–2 WIF_SCENARIO_TBL

Column Name	Description	Data Type	Maximum Length	Nullable (Y/N)
Scenario_ID	Identification number for a scenario, which is generated by the Calculation Engine.	Integer	32	N
New_Blackout_ End	No new markdown recommendations can be made before this date.	Date in format YYYY-MM-DD	10	Y
New_Inventory	The sum of inventory on hand, inventory on order, and, optionally, inventory in the warehouse. This value overrides total inventory quantity at end of history.	Integer	32	Y
New_Out_Dt	This value overrides BRPM's OUT_DT.	Date in format YYYY-MM-DD	10	Y
New_Inventory_ Target	This value overrides BRPM's INVENTORY_TARGET.	String	100	Y
New_Salvage_ Above_Target	This value overrides BRPM's SALVAGE_ABOVE_TARGET.	String	100	Y

WIF_ITEM_MARKDOWN_TBL - contains the markdown recommendation for a given Scenario_ID and item_ID.

Table 5–3 WIF_ITEM_MARKDOWN_TBL

Column Name	Description	Data Type	Maximum Length	Nullable (Y/N)
Scenario_ID	Identification number for a scenario, which is generated by the Calculation Engine.	Integer	32	N
Item_ID	Identifies the item.	Integer	32	N
Calendar_dt	The date for the markdown.	Date in format YYYY-MM-DD	10	N
Interpretation	PERM = 1 TEMP = 2 POS = 3	Integer	1	N
Relative_ Price	Markdown relative value.	Decimal	20,18	Y

WIF_RESULTS_TBL - provides a mapping between Scenario_ID/Item_ID and the Forecast_ID.

Table 5–4 WIF_RESULTS_TBL

Column Name	Description	Data Type	Maximum Length	Nullable (Y/N)
Scenario_ID	Identification number for a scenario, which is generated by the Calculation Engine.	Integer	32	N

Table 5–4 (Cont.) WIF_RESULTS_TBL

Column Name	Description	Data Type	Maximum Length	Nullable (Y/N)
Item_ID	Identifies the item.	Integer	32	N
Forecast_ID	Identifies the forecast.	Integer	32	N

What If and Inference Rules

What If is used to perform a recalculation of the optimization on a select group of items. Users can override certain settings to simulate changes using What If. Each re-optimization session is assigned a Scenario_ID by the Calculation Engine. The Scenario_ID is used to identify the specific What if calculation. The Scenario_ID is also used in all inference rules that are called during a What If calculation.

Scenario overrides (that is, the new values being used in the What If simulation) in What If are implemented just under the inference rule level so that the inference rules that are affected by What If can pick up the override values.

The following table details the relationship between the scenarios settings and specific inference rules:

Table 5–5 Scenario Settings in relationship to Inference Rules

Scenario Setting	Inference Rules
Exit Date	IR_ITEM_DATES, IR_ITEM_DATES_C, other inference rules that reference IR_ ITEM_DATES
Scenario Markdowns	IR_PENDING_MARKDOWNS
Inventory Level	IR_ACTIVITY_DATA
Inventory Target	IR_BUSINESS_POLICY (INVENTORYTARGET and TARGETSELLTHRU) (corresponds to BRPM values)
Salvage Value	IR_BUSINESS_POLICY (SALVAGEVALUEABOVETARGET)
End Blackout Date	IR_MARKDOWN_CALENDAR, IR_ MARKDOWN_CALENDAR_C

In addition, IR_PENDING_MARKDOWNS obtains markdowns from WIF_ITEM_ MARKDOWN_TBL, which contains the markdowns from the What If recalculation.

How What If Affects Inference Rules

Inference rules are affected by What If in one of three ways:

- Inference rules that have a column for Scenario_ID and contain override logic so that they can pick up the override values from WIF_SCENARIO_TBL:
  - IR_BUSINESS_POLICY
  - IR_ITEM_DATES
  - IR_ITEM_DATES_C
  - IR_MARKDOWN_CALENDAR
  - IR_PENDING_MARKDOWNS

    - IR_ACTIVITY_DATA
- Inference rules that have a column for Scenario_ID but no override logic. These inference rules have dependencies on the inference rules that do contain override logic:
    - IR_BLOCKED_MARKDOWN
    - IR_BLOCKED_MARKDOWN_C
    - IR_MARKDOWN_CALENDAR_EX
    - IR_MISSING_WEEKS
    - IR_PAST_TICKET_PRICES
    - IR_PLANNED_PROMOS
    - IR_PRICE_LADDER
    - IR_PRICE_LADDER_C
- Inference rules that specify Scenario_ID = 0 in internal queries so that they only retrieve optimization run data and not What If override data from other inference rules they have dependencies with.
    - IR_DISPLAY_PROMOS
    - IR_FORECAST_METRICS_POSTRUN
    - IR_ITEM_INFO
    - IR_ITEM_INFO_C
    - IR_ITEM_PRICES
    - IR_ITEM_PRICES_C
    - IR_PROJ_MARKDOWNS

Inference Rule Dependencies for What If

The scenario overrides from What if affect a number of inference rules directly and many others indirectly. Other linkages are possible, as described in the text. However, when you are configuring the inference rules, you should try to avoid breaking any of the linkages shown here, or incorrect What If behavior may result.

The following table lists the inference rule dependencies for What If. Inference rules are identified as primary in this table simply in terms of the dependency relationships being specified.

Table 5–6 Inference Rule Dependencies

Primary Inference Rule	Inference Rule(s) That Depend(s) on the Primary Inference Rule
IR_ITEM_DATES (_C)	IR_ACTIVITY_DATA
	IR_PENDING_MARKDOWNS
	IR_MISSING_WEEKS
	IR_PAST_TICKET_PRICES
	IR_MARKDOWN_CALENDAR (_EX)
	IR_BLOCKED_MARKDOWN(_C)
	IR_FORECAST_METRICS_POSTRUN
	IR_PROJ_MKDNS
	IR_HISTORIC_METRICS
	IR_DISPLAY_PROMOS
	IR_ITEM_INFO(_C)
	IR_PLANNED_PROMOS
IR_ITEM_PRICES(_C)	IR_PRICE_LADDER(_C)
	IR_HISTORIC_METRICS
	IR_DISPLAY_PROMOS
	IR_ITEM_INFO(_C)
	IR_PLANNED_PROMOS
IR_BUSINESS_POLICY	IR_PRICE_LADDER(_C)
	IR_MARKDOWN_CALENDAR
	P4P_WHAT_IF_ITEM_BASE
IR_WAREHOUSE	IR_ACTIVITY_DATA
	IR_BUSINESS_POLICY
	IR_FE_WAREHOUSE
	IR_HISTORIC_METRICS
IR_MISSING_WEEKS	IR_ACTIVITY_DATA
	IR_PAST_TICKET_PRICES
IR_ITEM_IDS(_C)	IR_PRICE_LADDER(_C)
	IR_MODEL_VALUES
IR_MARKDOWN_CALENDAR_EX	IR_MARKDOWN_CALENDAR
	IR_BLOCKED_MARKDOWN(_C)
IR_PENDING_MARKDOWNS	IR_ITEM_PRICES(_C)
IR_PLANNED_PROMOS	WIF_FORECAST_DATA
IR_METRICS	IR_SEASON_METRICS

What If Metric Calculations

The following tables list the metric calculations used by What If. Definitions for some terms used here can be found at the end of this section. Note that markupPercent has been added and gmDollarsRetail and gmPercentRetail have been removed.

The Opt Ticket Price reflects model run recommendations.

Table 5–7 Opt Ticket Price Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	forecastRecommendedTicket
End of Life (EOL)	NA
Total Till Out of Stock (TTOOS)	NA
Season to Date/Life to Date (STD/LTD)	NA
Weekly (corresponds to date in function) Note: Ticket prices are inventory weighted, not sales weighted.	sum for all items (getItemTicketPrice(item, date) * getSalesUnits(item, date))/ sum(getSalesUnits(item/date))
Monthly	NA

The Opt Sales Price reflects model run values, which will be different than the recommended ticket price only in the case of promotions.

Table 5–8 Opt Sales Price Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	forecastPriceAverage
End of Life (EOL)	NA
Total Till Out of Stock (TTOOS)	getTTOOSSalesDollars() / getTTOOSSalesUnits()
Season to Date/Life to Date (STD/LTD)	NA
Weekly (corresponds to date in function)	getForecastPriceAverage(): getSalesDollars(date) / getSalesUnits(date). Use getForecastPriceExact() if getSalesUnit=0
Monthly	getMonthSalesDollars(monthName) / getMonthSalesUnits(monthName)

The Ticket Price is a recalculation based on the user's overrides.

Table 5–9 Ticket Price Metric Calculation

Category/time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	forecastTicketPrice
End of Life (EOL)	NA
Total Till Out of Stock (TTOOS)	NA
Season to Date/Life to Date (STD/LTD)	NA
Weekly (corresponds to date in function)	Ticket price from Recalc Note: Ticket prices are inventory weighted, not sales weighted.

Table 5–9 (Cont.) Ticket Price Metric Calculation

Category/time Period	Tag Value/Calculation
Monthly	NA

The Sales Price is a recalculation based on the user's overrides, and is different than the ticket price in the case of temporary POS markdowns and promos.

Table 5–10 Sales Price Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	whatIfPrice2
End of Life (EOL)	NA
Total Till Out of Stock (TTOOS)	NA
Season to Date/Life to Date (STD/LTD)	NA
Weekly (corresponds to date in function)	Sales Price From Recalc Note: Sales prices are sales weighted, not inventory weighted.
Monthly	NA

The Sales Dollars represents the sum of the sales dollars for all items for a specified time period.

Table 5–11 Sales Dollars Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	salesDollars
End of Life (EOL)	getEOLSalesDollars(): getTTOOSSalesDollars() + getSTDSalesDollars();
Total Till Out of Stock (TTOOS)	getTTOOSSalesDollars(): Sum for all items for all dates till out date of sales dollars
Season to Date/Life to Date (STD/LTD)	getSTDSalesDollars(): Sum of (CUMULATIVE_ SALES_DOLLARS from item data for each item)
Weekly (corresponds to date in function)	getSalesDollars(date): Sum for all items (getSalesDollars(itemId,date)) getSalesDollars(item,date): (getItemSalesPrice(itemId, date) * getSalesUnits(itemId, date));
Monthly	getMonthSalesDollars: sum of getSalesDollars(date) for each item for each fiscal week

The Sales Units are the total sales units for the specified time period.

Table 5–12 Sales Units Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	salesUnits/ forecastSalesUnits

Table 5–12 (Cont.) Sales Units Metric Calculation

Category/Time Period	Tag Value/Calculation
End of Life (EOL)	getEOLSalesUnits()= getTTOOSSalesUnits() + getSTDSalesUnits()
Total Till Out of Stock (TTOOS)	getTTOOSSalesUnits(): sum (getSalesUnits(item,date)) for each item for each date till outdate of item
Season to Date/Life to Date (STD/LTD)	getSTDSalesUnits(): CUMULATIVE_ QUANTITY_ SOLD from item data for item
Weekly (corresponds to date in function)	getSalesUnits(item,date) : min(startingInventory,(item, date), sales(item,date)) sales(item,date)= Old Code: forecastDemand * Math.pow(basePrice, elasticity) / Math.pow(salesPrice, elasticity) New Code: Engine Recalc Sales Units
Monthly	getMonthSalesUnits: sum of getSalesUnits(item,date) for each item for each fiscal week

The Gross Margin Dollars (GM $) = Sales $ - (Unit Cost * Total Units)

The Adjusted Gross Margin Dollars (Adj GM $) = Sales $ - (Unit Cost * Total Units) + Residual Value of unsold units - Cost of unsold units

Unlike adjusted gross margin metrics, gross margin metrics are available on a weekly and monthly basis.

Table 5–13 GM$ Cost Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	gMDollarsCost
End of Life (EOL)	getEOLGrossMarginDollarsCost() :getTTOOSGrossMarginDollarsCost() + getSTDGrossMarginDollarsCost()
Total Till Out of Stock (TTOOS)	Remove from UI - for calc of EOL only) getTTOOSGrossMarginDollarsCost(): getTTOOSSalesDollars(item) - getTTOOSCostDollars(item) - getTTOOSEndingInventoryDollars Cost(item) + getSalvage(item) getTTOOSCostDollars(item): (getUnitCost(item) * getTTOOSSalesUnits(item)) getSalvage(itemid)= salvageValue * getTTOOSEndingInventoryUnits(item)
Season to Date/Life to Date (STD/LTD)	NA
Weekly (corresponds to date in function)	This calculation has been removed.

Table 5–13 (Cont.) GM$ Cost Metric Calculation

Category/Time Period	Tag Value/Calculation
Monthly	This calculation has been removed.

The Gross Margin Percent (GM %) Cost = GM $ / Sales $

The Adjusted Gross Margin% (Adj GM%) = Adj GM $/Sales $

Unlike adjusted gross margin metrics, gross margin metrics are available on a weekly and monthly basis.

Table 5–14 GM % Cost Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	gMPercentCost
End of Life (EOL)	(100.0 * getEOLGrossMarginDollarsCost()) / getEOLSalesDollars();
Total Till Out of Stock (TTOOS)	This calculation has been removed.
Season to Date/Life to Date (STD/LTD)	This calculation has been removed.
Weekly (corresponds to date in function)	This calculation has been removed.
Monthly	This calculation has been removed.

The Markup Percent = (Ticket Price - Unit Cost)/Ticket Price.

Table 5–15 MU % Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	markupPercent
End of Life (EOL)	NA
Total Till Out of Stock (TTOOS)	NA
Season to Date/Life to Date (STD/LTD)	NA
Weekly (corresponds to date in function)	getMarkupPerc(date)= 100*getMarkup(date) /getForecastTicketPrice(date) getMarkup(date) = sum for all items (getItemTicketPrice(item,date) - getUnitCost(item)) weighted by sales units
Monthly	NA

The Gross Margin Dollars Retail calculation has been removed.

Table 5–16 GM Dollars Retail Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	gMDollarsRetail
End of Life (EOL)	NA
Total Till Out of Stock (TTOOS)	This calculation has been removed.

Table 5–16 (Cont.) GM Dollars Retail Metric Calculation

Category/Time Period	Tag Value/Calculation
Season to Date/Life to Date (STD/LTD)	This calculation has been removed.
Weekly (corresponds to date in function)	This calculation has been removed.
Monthly	This calculation has been removed.

The Gross Margin Percent Retail calculation has been removed.

Table 5–17 GM Percent Retail Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	gMPercentRetail
End of Life (EOL)	NA
Total Till Out of Stock (TTOOS)	This calculation has been removed.
STD/LTD	This calculation has been removed.
Weekly (corresponds to date in function)	This calculation has been removed.
Monthly	This calculation has been removed.

The Markdown Dollars represents the forecasted markdown cost for the specified time period. The markdown cost is the sum of the costs based on permanent markdowns.

Table 5–18 Markdown Dollars Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	markdownDollars
Total Till Out of Stock (TTOOS)	getTTOOSMarkdownDollars(): sum of getMarkdownDollars() for all dates.
Season to Date/Life to Date (STD/LTD)	This calculation has been removed.
Weekly (corresponds to date in function) Note: This accounts for the cost of permanent markdowns only.	getMarkdownDollars()=sum(lastTicketPrice - currentTicketPrice) * getStartingItemInventory(itemId, date) of all items This calculation now uses the OWNED_RTL_PRICE to calculate the Ticket Prices as follows: TicketPrice = min(TicketPrice, OwnedPrice)
Monthly	sum of getMarkdownDollars (date) for each item for each fiscal week.

The Discount Dollars represents the forecasted discount cost for the specified time period. The discount cost is the sum of permanent and temporary markdown costs as well as promotions.

Table 5–19 Discount Dollars Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	discountDollars
Total Till Out of Stock (TTOOS)	getTTOOSDiscountDollars(): sum of getDiscountDollars() for all dates.

Table 5–19 (Cont.) Discount Dollars Metric Calculation

Category/Time Period	Tag Value/Calculation
Season to Date/Life to Date (STD/LTD)	This calculation has been removed.
Weekly (corresponds to date in function) Note: This accounts for permanent and temporary markdowns as well as promotions.	getDiscountDollars()=sum(lastTicketPrice - currentTicketPrice) * startingInventory+ Math.max(currentTicketPrice - currentSalesPrice, 0.0)* currentSalesUnits) of all items Ticket Prices calculated with respect to OWNED_RTL_PRICE. The Math.max ensures that negative Discount $ are not calculated.
Monthly	sum of getDiscountDollars(date) for each item for each fiscal week.

The Sell Thru Percent Remaining represents the amount of inventory remaining at the end of a specified time period.

Table 5–20 Sell Thru % Remaining Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	sellThruPercentRemaining
End of Life (EOL)	This calculation has been removed.
Total Till Out of Stock (TTOOS)	((100.0 * getTTOOSSalesUnits()) / getInitialInventory())getInitialInventory(): sum of INT_INVENTORY for all items.
Season to Date/Life to Date (STD/LTD)	This calculation has been removed.
Weekly (corresponds to date in function)	(100.0 * getSalesUnits()) / (getSalesUnits() + getEndingInventory())
Monthly	(100.0 * getMonthSalesUnits()) / (getMonthSalesUnits() + getMonthEndingInventory())

The Sell Thru Percent Total represents the total amount of inventory sold at the end of a specified time period.

Table 5–21 Sell Thru % Total Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	sellThruPercentTotal
End of Life (EOL)	getEOLSellThroughPercent
Total Till Out of Stock (TTOOS)	((100.0 * getTTOOSSalesUnits()) / getTotalInventory())
Season to Date/Life to Date (STD/LTD)	(100.0 * getSTDSalesUnits(item)) / (getSTDSalesUnits(item) + getSTDEndingInventoryUnits()))
Weekly (corresponds to date in function)	(getSalesUnits(date) * 100) / getTotalInventory()getTotalInventory(): sum (getCumulativeSales(itemId) + getInitialInventory(itemId)) of all items
Monthly	getSellThruPercentTotal(date) for each item for each fiscal week

The EOP Units are the remaining units on hand at the end of a specified time period.

Table 5–22 EOP Units Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	inventoryUnits/forecastInventoryUnits
End of Life (EOL)	getTTOOSEndingInventoryUnits
Total Till Out of Stock (TTOOS)	getTTOOSEndingInventoryUnits:sum of getEndingInventory(itemId, outdate) for each item
Season to Date/Life to Date (STD/LTD)	getSTDEndingInventoryUnits(): sum of (COMMITTED_INV_UNITS from item data) for all items
Weekly (corresponds to date in function)	getEndingInventory(date): sum (getEndingInventory(item,date)) for all itemsgetEndingInventory(item,date):getStartingItemInventory(item,date) - getSalesUnits(item,date)
Monthly	getMonthEndingInventory():sum of getEndingInventory(date) for each item for each fiscal week

The EOP Dollars (Cost) represents the cost of the remaining units on hand.

Table 5–23 EOP Dollars (Cost) Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	inventoryDollarsCost
End of Life (EOL)	getTTOOSEndingInventoryDollarsCost
Total Till Out of Stock (TTOOS)	getTTOOSEndingInventoryDollarsCost(item) : getUnitCost * getEndingInventory(itemId, outdate)
Season to Date/Life to Date (STD/LTD)	Sum (getUnitCost(item) * getSTDEndingInventoryUnits(item)) for allitems
Weekly (corresponds to date in function)	sum (getUnitCost(itemId) * getEndingInventory(itemId, date)) for all items
Monthly	getEndingInventoryDollarsCost(date) for each item for the fiscal month (notice that this is not a sum)

The EOP Dollars (Retail) represents the retail value of the remaining units on hand.

Table 5–24 EOP Dollars (Retail) Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	inventoryDollars/inventoryDollarsRetail
End of Life (EOL)	getTTOOSEndingInventoryDollarsRetail
Total Till Out of Stock (TTOOS)	getTTOOSEndingInventoryDollarsRetail(item):sum of (getEndingInventoryDollarsRetail(item. outdate))
Season to Date/Life to Date (STD/LTD)	sum (ticket price *getSTDEndingInventoryUnits(item)) for all items

Table 5–24 (Cont.) EOP Dollars (Retail) Metric Calculation

Category/Time Period	Tag Value/Calculation
Weekly (corresponds to date in function)	sum (getOwnedSalesPrice(itemId, date) * getEndingInventory(item,date)) for all items
Monthly	getEndingInventoryDollarsRetail(date) for each item for the fiscal month (notice that this is not a sum)

The Promo Flag indicates whether a planned promotion is in effect or not for a given period of time.

Table 5–25 Promo Flag Metric Calculation

Category/Time Period	Tag Value/Calculation
Tag Used in config/Price/grids/p4pgui-config.xml	promoFlag
End of Life (EOL)	NA
Total Till Out of Stock (TTOOS)	NA
Season to Date/Life to Date (STD/LTD)	NA
Weekly (corresponds to date in function)	P if there is a promo in week.
Monthly	P if there is a promo in month.

Metric Table Definitions

The following are definitions used in the tables in this section. Also listed are some Forecasted End of Life (FCEOL) metric calculations.

- Cumulative Sales Units: The total sales through the period.
- Discount $: The forecasted discount cost for the specified time period. The discount cost includes only temporary markdown costs.
- EOP Units: The forecasted remaining units at the end of the period.
- EOP $ (Cost): The forecasted cost of the remaining units at the end of the period.
- EOP $ (Retail): The forecasted retail value of the remaining units at the end of the period.
- EOL: End of Life. The forecasted total from the beginning of the life cycle to the exit date, using the forecast and markdowns from the weekly run.
- FCEOL: Forecasted End of Life. The forecasted total from the beginning of the life cycle to the exit date, using the forecast and markdowns from the weekly run.
- GM $: Gross Margin $ = Sales $ - (Unit Cost * Total Units) + (Residual value of unsold units)
- GM %: Gross Margin % = GM $/Sales $
- LTD: Life to Date. The total from the beginning of the life cycle to the most recent week of history.
- MD $: The forecasted markdown cost for the specified time period. The markdown cost is the sum of the costs based on permanent markdowns.
- MU %: Markup % = (Ticket Price - Unit Cost)/Ticket Price.

- Opt Sales Price: The forecasted sales price for the week, assuming all recommended markdowns are taken. This price can be lower than the forecasted ticket price due to promotions. It can be higher than the ticket price in markdown weeks since some sales in those weeks may be forecasted to occur prior to the markdown.
- Opt Ticket Price: The ticket price forecasted during the weekly model run for the end of a specific period.
- Sales $: The total sales dollars for the specified time period.
- Sales Price: The sales price resulting from the what-if simulation. In weeks with markdowns, this number may be higher than the ticket price if any sales occurred at the pre-markdown price.
- Sales Units: The total sales units for the specified time period.
- Sell Thru %: The percentage of units sold during a given period relative to units at the beginning of the period. It is calculated as Sales Units/(Sales Units + EOP Units).
- Sell Thru % (Total): The percentage of units sold during a given period relative to the total buy quantity. It is calculated as Sales Units/(Cumulative Sales Units + EOP Units).
- Ticket Price: The ticket price resulting from the what-if simulation. If no recalculation has occurred, then this will be the same as the Opt Ticket Price.
- TTOOS: Total Til Out of Stock. The forecasted total form the start of the simulation to the exit date. This displays the weekly model run's recommended scenario, initially or after a Show Initial Recommendations action. It also displays the forecast and markdowns form the what-if simulation if a recalculation has occurred.
- currentSalesPrice: sales price for item or week.
- currentTicketPrice: ticket price for this week.
- getInitialInventory(): INT_INVENTORY - column - committed_inv_units from item_data.
- getItemSalesPrice(itemid, date): sales price for item/date.
- Old code: sales price based on user selection
- New code: engine recalc sales price
- getOriginalRetailPrice(): original retail from item data weighted by sales units for multiple items.
- getOwnedPrice(ticketPrice, itemID): Market value of item. It is used to calculate MD $ and Retail $ and Discount $.
- Math.min(ticketPrice, owned Price)
- getStartingItemInventory(itemid, date): starting inventory for item for this week.
- getUnitCost(): unit cost for item from item data.
- lastTicketPrice: get the lowest ticket price iterating through all the previous weeks' data.

FCEOL Metrics

The following are the Forecasted End of Life (FCEOL) metric calculations:

- FCEOL Sales$ = EOL_CUM_DOLLARS_SALES + CUMULATIVE_SALES_ DOLLARS from p4pgui_display_items (projSalesDollarsEOL - p4p-column-list.xmly)
- FCEOL Sales Units = EOL_CUM_UNIT_SALES + CUMULATIVE_QUANTITY_ SOLD from p4p_display_items (projSalesUnitEOL - p4p-column-list.xmly)
- FCEOL GM$ = (PROJ_STC_EOL_GM_AMOUNT form p4p_display_items (projMarginDollarsEOL - p4p-column-list.xmly)
- FCEOL EOP Units = ENDING_INVENTORY_UNITS from p4p_display_items (projOnHandUnitsEOL - p4p-column-list.xmly)
- FCEOL EOP $ Retail = (case when process_as_itme = 1 then ending_inventory_ units else ending_inventory_units_c end)*REC_RTL_MIN from p4p_display_items (projOnHandRtlDollarsEOL - p4p-column-list.xmly)

The remaining FCEOL metrics are the same as EOL for calculations but are always based on the model run results.

6

Configurable Data Attributes

CDAs are used to stage and load customized data.

This chapter contains the following sections:

Introduction

Configurable Data Attributes (CDAs) provide a way for retailers to see, in addition to the default data that is visible through the application interface, custom data that they themselves specify and that is not required by the application. This data can be used in business rules and can be displayed in the application UI.

Defining Configurable Data Attributes

Configurable Data Attributes are defined in the database using the CDA Administration Utility. The data is then staged and loaded. All retailer-specified data is included in the standard interface specification in fields with field names beginning with the word ATTRIBUTE.

CDAs are disabled by default. The column PL_DD_ATTRIBUTES.DISABLED should be set to 1 to disable the CDA and should be set to 0 to enable the CDA.

You can access the CDAs in the database using database queries or change the grid configuration to make them visible in the user interface.

The number of CDAs per entity is limited by the number of database columns pre-allocated in every CDA storage table. Every application schema provides eight data columns of type VARCHAR and DATE, and ten number columns of type NUMBER. When you are creating a new attribute, you can choose the storage columns from the following disassociated columns of the corresponding type:

Table 6–1 CDA Data Type

Data Attribute Type	Data Type
String	VARCHAR
Integer	NUMBER
Boolean	NUMBER
Double	NUMBER
Date	NUMBER

Table 6–1 (Cont.) CDA Data Type

Data Attribute Type	Data Type
Currency	VARCHAR
Currency	NUMBER (2 columns)

The following tables supports extension by the CDA Administrative Utility:

Table 6–2 Standard Interface Tables with CDAs

Entity Name	Staging Table	Active Table	CDA Table
Location	ASH_LH_TBL	LOCATION_HIERARCHY_TBL	LH_CDA_TBL
Merchandise	ASH_MH_TBL	MERCHANDISE_HIERARCHY_TBL	MH_CDA_TBL

7

Internationalization

MDO is translated into multiple languages. The details of internationalization are discussed in this chapter.

This chapter contains the following sections:

Introduction

Internationalization is the process of creating software that can be translated more easily. Changes to the code are not specific to any particular market. MDO has been internationalized to support multiple languages.

This chapter describes configuration settings and features of the software that ensure that the base application can handle multiple languages.

Translation

Translation is the process of interpreting and adapting text from one language into another. Although the code itself is not translated, components of the application that are translated include the following:

- Graphical user interface (GUI)
- Error messages

The following components are not translated:

- Documentation (Online Help, Release Notes, Installation Guide, User Guide, Operations Guide)
- Batch programs and messages
- Log files
- Configuration Tools
- Reports

- Demonstration data
- Training Materials

The user interface for MDO has been translated from U.S. English into the following languages:

- Chinese (Traditional)
- Chinese (Simplified)
- Croatian
- Dutch
- French
- German
- Greek
- Hungarian
- Italian
- Japanese
- Korean
- Polish
- Portuguese (Brazilian)
- Russian
- Spanish (Spain)
- Swedish
- Turkish

Files

The following Markdown Optimization files are translated for each language that is supported by an installation. These files are the resource files used by the application for culturally dependent data and text strings that are displayed to end users. A properties file may exist in more than one subdirectory; any changes must be made consistently in all versions of each properties file. These files do not have to be registered with the application server.

Translated Files

The translated files each exist in two different formats.

- The format used by the application (ending in .properties)
- The format that can be used for customization (ending in .native)

Note that the _xx in the filename designates the locale of the file. The locale can be the language alone (e.g., _en, _fr), or a language_country combination (e.g., _en_GB, _fr_FR). Refer to the "Supported Locales" section of the Java Internationalization documentation appropriate for the version of Java that you are using.

```
./config/Price/resources/formats_xx.properties
./config/Price/resources/formats_xx.properties.native
.
.
```

```
./config/Price/resources/gridResources_xx.properties
./config/Price/resources/gridResources_xx.properties.native
.
.
./config/Price/resources/p4pguiResources_xx.properties
./config/Price/resources/p4pguiResources_xx.properties.native
.
.
./config/Price/resources/UserMessageResources_xx.properties
./config/Price/resources/UserMessageResources_xx.properties.native
.
.
./config/suite/resources/businessrulemgrResources_xx.properties
./config/suite/resources/businessrulemgrResources_xx.properties.native
.
.
./config/suite/resources/CommonMessages_xx.properties
./config/suite/resources/CommonMessages_xx.properties.native
.
.
./config/suite/resources/EngineResources_xx.properties
./config/suite/resources/EngineResources_xx.properties.native
.
.
./config/usermanagement/resources/gridResources_xx.properties
./config/usermanagement/resources/gridResources_xx.properties.native
.
.
./config/usermanagement/resources/UsermanagementResources_xx.properties
./config/usermanagement/resources/UsermanagementResources_xx.properties.native
.
.
./config/usermanagement/UserAccount_xx.properties
./config/usermanagement/UserAccount_xx.properties.native
```

Directory Structure

Beneath the configuration root directory is the Markdown Optimization application root directory. This directory contain subdirectories for default resources and default grid configurations. The application root directory also contains the retailer-specific configuration directory. The Client directory contains the properties files that have been localized/configured for a retailer implementation.

Files that are localized are named according to the following convention, using the base name of the file and adding a language specification, as shown in the following example. Note that, since two dialects of Chinese are supported, the part of the string that identifies the language contains two parts. This pattern will apply to any language in which more than one dialect is supported.

Language Name gridResources_xx.properties

The localized files contain the translated user interface strings, localized date and number formats, but no locale-insensitive information such as database or installation configuration properties.

Configuration Settings

In order for Markdown Optimization to function correctly when localized, the end user's Browser settings and the Regional and Language Options settings of the

operating system must match. If they do not match, the UI may display in an inconsistent manner. The Browser settings can be found in the Internet Explorer under Tools > Internet Options > Languages. The Regional and Language Options can be found in the Control Panel.

The following table lists these settings:

Table 7–1 Language Settings

Browser Settings	Regional and Language Option
Chinese (China) [zh-CN]	Chinese (PRC)
Chinese (Taiwan) [zh-TW]	Chinese (Taiwan)
Croatian [hr-HR	Croatian
Dutch [nl-NL]	Dutch
English (United States) [en-us]	English (United States)
French (France) [fr-FR]	French (France)
German (Germany) [de]	German (Germany)
Greek [el-GR]	Greek
Hungarian hu-HU	Hungarian
Italian [it-IT]	Italian
Japanese [ja-JP]	Japanese
Korean [ko-KR]	Korean
Polish [pl-PL]	Polish
Portuguese (Brazil) [pt-BR]	Portuguese (Brazil)
Russian [ru-RU]	Russian
Spanish (International Sort) [es-ES]	Spanish (Spain)
Swedish	Swedish
Turkish [tr-TR]	Turkish

Formatting for Internationalization

The formatting of dates, time, and numbers is locale-specific and determined by the Browser settings and the Regional settings.

Currency

Note: A single MDO installation can be configured to support more than one underlying currency. Typically, items would be segregated into worksheets by currency regions. However, MDO only supports a single currency symbol used for formatting tabular financial data within a single implementation. A separate currency metric column or other labels can be configured to convey which currency is appropriate for a given table.

To set the currency symbol for the application, edit CommonMessages.properties.

Specifically, you must edit CommonMessages_*xx*.properties.native, using the procedure described below, to update your edits into CommonMessages_ xx.properties.

For example, to specify the Euro in French, edit CommonMessages_fr.properties.native as follows:

```
cc.currencySymbol.localOverride=€
```

While you can edit the CommonMessages_fr.properties directly and set the value to the symbol for the Euro, if you need to update other strings you will overwrite the Euro setting and lose it unless you have updated the change to the native file.

Format Patterns

Number format patterns are used to specify the arrangement of digits, group and decimal separators, percent symbols, and currency symbols in numeric formats. Date format patterns specify the arrangement of seconds, minutes, hour, day, month, year, and day of week and date separators in date formats. In Markdown Optimization, the format patterns are always specified using US English separator conventions. The currency symbols themselves and the numeric separator character are not specified in the format patterns.

Default format patterns are specified in CommonMessages.properties. These can be edited to match retailer and or locale requirements. Format patterns can be found in almost any of the properties files, but they are most common in gridResouces.properties.

Number Format Patterns

Number format patterns are expressed with US English separators ("comma" for the thousands separator and always "period" for the decimal separator), regardless of the locale. For example, #0.00%, 0 %, and #.0000 % are all valid formats. A specific configuration may require that in English the user sees two decimal places and in European Union the user sees one decimal place. The pattern string for English is "#0.00%" and the pattern string for French is "% #0.0". The user sees 59.29 % in English and % 59,3 in French.

The pattern string "#0,00" is an invalid format because it uses the comma as the decimal separator. It should be #0.00.

Currency Format Patterns

Currency format patterns behave the same as number format patterns except that they contain the universal currency symbol to show the position of the currency symbol "¤" in the format. This symbol is represented as \u00A4.

For example #,##0.00 \u00A4 ; (#,##0.00 \u00A4) is a reasonable format for an EU country using the Euro. If the currency symbol is set to be the Euro as describe above, the following value will be formatted as follows for the German locale:

```
-123456.99 -> (123 456.99 €)
.99 -> 0.99 €
```

Date Format Patterns

Date Format Patterns must always be expressed in US English symbols.

Table 7–2 Date Format Patterns

Letter	Date/Time Component	Presentation	Example
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	AM/PM marker	Number	PM
H	Hour in day (0 - 23)	Number	024
k	Hour in day (1 - 24)	Number	0
K	Hour in AM/PM (0 - 11)	Number	12
h	Hour in AM/PM (1 - 12)	Number	30
m	Minute in hour	Number	55
s	Second in minute	Number	978
S	Millisecond	Number	
z	Time zone	General time zone	Pacific Standard time; PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800

For example, in Portugal, the day precedes the month, and dashes are the typical separator. Note that unlike the Number Format patterns, the separator characters are defined in the pattern. So in the Portuguese locale, a short date might be specified as follows:

dd-MM-yyyy

Number Separators

By default, Markdown Optimization displays thousands and decimal separators as determined by Java's internationalization framework. The default behavior of the application should match the locale of the user's browser.

For example, if the format pattern #,##0.00 is used for the floating point number 1234.56, it displays as follows, depending on the browser setting:

Table 7–3 Number Display

Browser Setting	Number Displayed
EN	1,234.56
NL	1.234,56
FR	1 234,56

To override the default, edit CommomMessages.properties. For example, to define specific separators:

```
cc.groupSeparatorSymbol.localeOvverride=.
cc.decimalSeparatorSymbol.localeoverride=,
```

This setting produces the following results:

Table 7–4 Number Display

Browser Setting	Number Displayed
EN	1.234,56
NL	1.234,56
FR	1.234,56

formats.properties

The formats.properties file contains the number and date formats used for insertion into some user-facing error messages. A file exists for each locale supported by Markdown Optimization. The separator symbols in the format strings must always be "comma" for the thousands separator and must always be "period" for the decimal separator, regardless of the locale.

For example, #0.00%, 0 %, and #.0000 % are all valid formats. A specific configuration may require that in English the user sees two decimal places and in EU the user sees one decimal place. The pattern string for English is "#0.00%" and the pattern string for French is "% #0.0". The end user sees 59.29 % in English and % 59,3 in French.

The pattern string "#0,00" is an invalid format because it uses the comma as the decimal separator.

Technical Notes

The cutoff sets the time and day of week for the cutoff, which is displayed in the application UI.

The entry-cutoff-day-of-week and export-day-of-week attributes of the cutoff tag must use English day-of-week names.

Markdown Optimization depends on both the browser settings and the regional settings to determine which language is being supported for a specific implementation.

8

Pricing Group Management

This chapter provides details about pricing group configuration.

It contains the following sections:

Introduction

Pricing groups (previously called "collections") in Markdown Optimization are traditionally managed at the optimization level (non-chain pricing groups). Markdown Optimization also permits pricing groups to be managed at the chain level but optimized at a lower level.

Chain level pricing groups can be useful, for example, when adding merchandise to a pricing group in all locations. Instead of adding the merchandise to each location separately, a user can add the merchandise only once, at the chain level, and the merchandise will be added by the application to each location. Although the pricing group is managed at the chain level, the worksheet displays the pricing group at the optimization level to facilitate taking markdowns.

Two inference rules provide configuration points for pricing groups. The IR_ITEM_COLLECTION inference rule is used to provide custom configuration for defining what is included or excluded from the pricing group load. The IR_COLLECTION_OPTION inference rule contains a flag that is used to indicate whether pricing groups are managed at the chain level or at the optimization level. Note that IR_COLLECTION_INFO continues to be used for optimization without regard to pricing group management.

Three load procedures provide the functionality for loading pricing groups into Markdown Optimization: LoadCollectionsAuto, LoadCollectionsSendback, and LoadCollectionsFE.

Pricing Group information is stored in ITEM_DATA and in P4P_COLLECTIONS (both populated by LoadCollectionsFE). If pricing groups are managed at the chain level, the Collection_ID column in ITEM_DATA (which is used by the front end) is populated with a parent record, and the Internal_Collection_ID column (which is used by the model) is populated with children records, and P4P_COLLECTIONS is populated with chain-pricing group information.

If pricing groups are managed at the optimization level, both columns are populated with the optimization-level record, and P4P_COLLECTIONS is populated with item-pricing group information.

Load Procedures

There are three load procedures for pricing groups: LoadCollectionsAuto, LoadCollectionsSendback (both part of the Standard Load - run from pl_load_client.sh), and LoadCollectionsFE (part of FELOAD in load_statements.sql - run from plfrontendload.sh). None of these loads require ASH staging files. For more information on the standard load and the model run, see the *Oracle Retail Markdown Optimization Operations Guide*.

LoadCollectionsAuto

The ir_item_collection view determines how to group items into pricing groups, and the LoadCollectionsAuto procedure creates new pricing groups and new pricing group maps based on the view. All items with the same COLLECTION_CLIENT_ID are assigned to the same pricing group. If any item has already been assigned to a pricing group, or has already been auto-collected, the load procedure excludes it (using the ITEM_ASSIGNED_COLLS_TBL table).

The procedure populates COLLECTIONS_TBL with distinct pricing groups and populates COLLECTION_MAPS_TBL with the mappings for pricing groups to items. If an item is not already in COLLECTION_MAPS_TBL, it will be auto-collected as part of the load procedure.

The LoadCollectionsAuto procedure derives the rules for grouping items into pricing groups from the IR_ITEM_COLLECTION inference rule. See "Inference Rule Configuration" on page 8-3 for details on configuring the inference rule.

A flag in IR_ITEM_COLLECTION_OPTION determines whether the pricing groups are managed at the chain level or at the optimization level. When the flag is set to N (the default value), pricing groups are managed at the optimization level. When the flag is set to Y, pricing groups are managed at the chain level.

To disable auto-collection, you can redefine IR_ITEM_COLLECTION so that it does not return any records (for example, by adding 1=0 to the where clause in the view).

New items that become eligible or ineligible are automatically added or removed from a chain level pricing group as long as they are part of the Items data feed.

LoadCollectionsSendback

Any changes or additions to pricing groups that users implement using the application UI override the assignment of items to pricing groups that are made by the LoadCollectionsAuto procedure. Changes are applied to the associated P4P_COLLECTIONS and ITEM_DATA entries (in the front end) and propagated to COLLECTION_TBL and COLLECTION_MAPS_TBL (in the back end). The population of these tables differs, depending on the setting in IR_ITEM_COLLECTION_OPTION.

The changes are processed by the LoadCollectionsSendback procedure and will be reflected in the UI the week after they are processed by the procedure. The modifications are archived in SENDBACK_COLLECTIONS_ARCH.

These changes include:

- Items re-assigned from one existing pricing group to another.

- Items re-assigned to a new pricing group.
- Items removed from an existing pricing group that are not assigned to another pricing group. If such items have been auto-collected, they cannot be auto-collected again, but must be manually assigned to another pricing group.

LoadCollectionsFE

The LoadCollectionsFE procedure is part of the FELOAD step (called by plfrontendload.sh) in load_statements.sql. It populates P4P_COLLECTION and updates the Collection_ID (used by the front end) and Internal_Collection_ID (used by the model) columns in ITEM_DATA. Source tables are COLLECTION_TBL and COLLECTION_MAPS_TBL.

Inference Rule Configuration

The IR_ITEM_COLLECTION inference rule defines how items are grouped into pricing groups. The IR_ITEM_COLLECTION_OPTION is set either to N, which indicates that the pricing groups are managed at the level of optimization, or to Y, to indicate pricing group management at the Chain level.

Example Configurations

Here are three sample configurations.

IR_ITEM_COLLECTION

Here is an example configuration of IR_ITEM_COLLECTION for pricing groups. The single point of configuration is the collection_client_id. This column defines parent pricing groups in the non-default (Y) implementation. It defines child pricing groups in the default (N) implementation by including Location_ID, as shown in the following example.

```
CREATE VIEW ir_item_collection
(ITEM_ID, MERCHANDISE_ID, LOCATION_ID
 COLLECTION_CLIENT_ID, COLLECTION_DESC)
AS
SELECT item_id,
       i.merchandise_id,
       i.location_id,
       case when iro.chain_flag='Y'then mh1.client_load_id||'/'||i.season_code
       else mh1.client_load_id||'/'||i.season_code||'/'||i.location_id end as
           collection_client_id,
       mh1.merchandise_desc || '/' ||i.season_code as collection_desc
FROM items_tbl i, merchandise_hierarchy_tbl mh, merchandise_hierarchy_tbl mh1, ir_
item_collection_option iro
       WHERE mh.merchandise_id = i.merchandise_id and mh1.merchandise_id =
           mh.parent_merchandise_id
```

Chain-Level Pricing Groups

An example of Chain Pricing Group flag (IR_ITEM_COLLECTION_OPTION):

```
CREATE VIEW ir_item_collection_option AS
       SELECT 'Y' AS chain flag from DUAL-- identifier of chain collection
management implementation
```

Redefining Pricing Groups

If you want to change the way items are grouped into pricing groups, you must do the following:

1. Update IR_ITEM_COLLECTION
2. Truncate the following tables:
   - COLLECTION_MAPS_TBL
   - COLLECTIONS_TBL
   - ITEMS_ASSIGNED_COLL_TBL
3. Re-run the LoadCollectionsAuto procedure. This occurs as part of the weekly batch process, so the new grouping will not be available until the next model run.

Front End Configuration

The application Maintaining Merchandise page should provide a grid in which the top level displays the chain pricing group and the level below it displays the name of the merchandise in the pricing group, independent of the location it belongs to.

The edit-group-grid.xml file must be configured so that a user can select merchandise and add it across all locations/regions.

Here is a grid configuration that exemplifies the above characteristics. Other configurations are possible. The configuration should be consistent with the rest of the application configuration.

```
 <row-group>
        <key>Style-CC</key>
        <rowgroup-properties expandable="false" isexpanded="true">
            <groupby>Style</groupby>
        </rowgroup-properties>
        <override-column-group>
            <column>
                <key>INT_STYLE_DESC</ key>
                <column-properties display-type="static-text"/>
            </column>
            <column>
               <key>ROWSELECT</key>
               <column-properties editable="true" display-type="checkbox"
                                  editable-in-readmode="true"/>
            </column>
        </override-column-group>
        <row-group>
             <key>CC-Cluster</key>
             <rowgroup-properties/>
        </row-group>
        <summary-info/>
</row-group>
```

It will change to:

```
<row-group>
        <key>Style-CC</key>
        <rowgroup-properties expandable="false" isexpanded="false">
            <groupby>Style</groupby>
        </rowgroup-properties>
        <override-column-group>
            <column>
```

```
                <key>INT_STYLE_DESC</ key>
                <column-properties display-type="static-text"/>
            </column>
        </override-column-group>
</row-group>
```

Pricing Group Functionality - An Example

Here is an example of how pricing groups can function:

1. Create a new pricing group with collection_id=3 in p4p_collection using the application UI.
2. Assign the following items, from ITEM_DATA, to the pricing group:
   - M1 - L1
   - M2 - L1
3. The item_tbl table should contain the following:
   - M1 - L1
   - M2 - L2
   - M2 - L3
   - M2 - L4
4. In the non-pricing group (default) implementation, collection_id = M#/S1/L1, includes the following items. Only one location ID can appear here. After the load, there is one collection_tbl record and three collection_maps_tbl records.
   - M1 - L1
   - M2 - L1
   - M3 - L1
5. The pricing group (non-default) implementation, collection_id = M#/S1, includes the following items. Since the pricing group client ID defines the parent pricing group, after the load, there is one parent collection_tbl.
   - M1 - L1
   - M2 - L1
   - M3 -L1
   - M1 - L2
   - M2 - L3
   - M3 - L3
6. Since there are three distinct locations selling different merchandise, there are three child collections_tbl records.

   The collection_maps_tbl records for child pricing group #1 include:
   - M1 - L1
   - M2 - L1
   - M3 - L1

   The collection_maps_tbl record for child pricing group #2 is M1 - L2.

   The collection_maps_tbl records for child pricing group #3 include:

- M2 - L3
- M3 - L3

7. To have an item unassigned from a pricing group, use the application UI to remove (set collection_id to NULL) for a given item. Then the sendback procedure will remove the record from collection_maps_tbl.

Test Scenarios - LoadCollectionsSendback

Here are five scenarios using LoadCollectionsSendback that describe the condition before the load procedure occurs and what happens as a result of the load procedure.

Scenario 1: Adding front end pricing groups and assigning items to the pricing groups. Back end tables are empty.

- Before the Load: There is one pricing group in p4p_collection. Four items in the ITEM_DATA table have been assigned to the pricing group. Both collections_tbl and collection_maps_tbl are empty.

-
col_id
1

ITEM_DATA Table

col_id	m_id	l_id
1	M1	L1
1	M1	L2
1	M2	L1
1	M2	L2

Items

m_id	l_id
M1	L1
M1	L2
M2	L1
M2	L2
M2	L3

- After the Load:
  - Default Single-Chain Implementation (chain_flag = N). One record is created in collections_tbl (regardless of the number of distinct locations for the four items) and four records are created in collection_maps_tbl.

Collections_tbl	
col_id	parent_col_id
1	null

Collection_maps_tbl		
col_id	m_id	l_id
1	M1	L1
1	M1	L2
1	M2	L1
1	M2	L2

- Non-Default Multi-Chain Implementation (chain_flag = Y). One record is created in collections_tbl with no maps. The number of child records created matches the number of distinct locations. Each child pricing group is associated with a collection_maps_tbl record that is created in association with an assigned item.

Collections_tbl	
col_id	parent_col_id
1	null
2 (for L1)	1
3 (for L2)	1
4 (for L3)	1

Collection_maps_tbl		
col_id	m_id	l_id
2	M1	L1
2	M2	L1
3	M1	L2
3	M2	L2
4	M2	L3

Scenario 2: Removing an item from a pricing group.

- Before the Load: The collection_id for on of the items in the ITEM_DATA table is set to NULL.

Collections_tbl
col_id

Collections_tbl
1
5
6

ITEM_DATA Table		
col_id	m_id	l_id
1	M1	L1
1	M1	L2
1	M2	L1
1 (set to null)	M2	L2
5 (set to null)	M6	L6
6 (set to null)	M5	L5

Items	
m_id	l_id
M1	L1
M1	L2
M2	L1
M2	L2
M2	L3
M6	L6
M5	L5
M5	L7

- After the Load:
  - Default Single-Chain Implementation (chain_flag = N). One pricing group map for the item is deleted. The collection_tbl record is also removed if no other items for the pricing group are found in the ITEM_DATA table and, for merchandise that does not exist in the ITEM_DATA table for the collection_id, no other items are found in pricing group maps.

Collections_tbl	
col_id	parent_col_id
1	null
5	null
6	null

Collection_maps_tbl		
col_id	m_id	l_id
1	M1	L1
1	M1	L2
1	M2	L1
1	M2	L2
5	M6	L6
6	M5	L5
6	M5	L7

- Non-Default Multi-Chain Implementation (chain_flag = Y). One or two pricing group maps for the item are deleted. (The child record is deleted as the second record if there are no other items in the ITEM_DATA table or in maps with that merchandise, but if the same merchandise has different locations and exists in maps.) The collection_tbl records (parent and child) are also removed if no other items for the pricing group are found in the ITEM_DATA table (regardless of whether there are more items with that merchandise in the items for the collection_id).

Collections_tbl	
col_id	parent_col_id
1	null
2 (for L1)	1
3 (for L2)	1
4 (for L3)	1
5	null
6	null
7 (for L6 only)	5
8 (for L5 only)	6
9 (for L7 only)	6

Collection_maps_tbl		
col_id	m_id	l_id
2	M1	L1
2	M2	L1
3	M1	L2
3	M2	L2
4	M2	L3
7	M6	L6

Collection_maps_tbl		
8	M5	L5
9	M5	L7

Scenario 3: Adding an item to a pricing group.

- Before the Load: An item in the ITEM_DATA table is assigned to an existing pricing group. The front end updates collection_id for the item.

p4p_collection
col_id
1

ITEM_DATA Table		
col_id	m_id	l_id
1	M1	L1
1	M1	L2
1	M2	L1
1	M2	L2
1	M3	L4

Items	
m_id	l_id
M1	L1
M1	L2
M2	L1
M2	L2
M2	L3
M3	L4

- After the Load:
  - Default Single-Chain Implementation (chain_flag = N). One pricing group map for the item is created.

Collections_tbl	
col_id	parent_col_id
1	null

Collection_maps_tbl		
col_id	m_id	l_id
1	M1	L1
1	M1	L2
1	M2	L1
1	M2	L2
1	M3	L4

- Non-Default Multi-Chain Implementation (chain_flag = Y). One child pricing group map is created for the item. If the merchandise being added is new to the ITEM_DATA table, then one parent map is created with chain location_id. If the location is new, then one child pricing group is also created.

Collections_tbl	
col_id	parent_col_id
1	null
2 (for L1)	1
3 (for L2)	1
4 (for L3)	1
5 (for L4)	1

Collection_maps_tbl		
col_id	m_id	l_id
2	M1	L1
2	M2	L1
3	M1	L2
3	M2	L2
4	M2	L3
5	M3	L4

Scenario 4: Adding a front end pricing group without assigning any items to it.

- Before the Load: A standalone record is created in p4p_collection when a user creates a new pricing group but does not assign any items to it.

p4p_collection
col_id
1
5

ITEM_DATA Table		
col_id	m_id	l_id
1	M1	L1
1	M1	L2
1	M2	L1
1	M2	L2

Items	
m_id	l_id
M1	L1
M1	L2
M2	L1
M2	L2
M2	L3

- After the Load:
  - Default Single-Chain Implementation (chain_flag = N). The p4p_collection record is deleted.

p4p_collection
col_id
1
5

  - Non-Default Multi-Chain Implementation (chain_flag = Y). The p4p_collection record is deleted.

p4p_collection
col_id
1
5

Scenario 5: Items are moved from one pricing group to another while adding a front end pricing group and assigning items to it. Back end tables contain items.

- Before the Load: A p4p_collection record is created. Items are assigned to the pricing group. Some items in the ITEM_DATA table are updated with the new collection_id.The back end tables contain other pricing group data.

Collections_tbl
col_id

Collections_tbl
1
5

ITEM_DATA Table		
col_id	m_id	l_id
1	M1	L1
1	M1	L2
1 - > 5	M2	L1
1 - > 5	M2	L2

Items	
m_id	l_id
M1	L1
M1	L2
M2	L1
M2	L2
M2	L3

- After the Load:
  - Default Single-Chain Implementation (chain_flag = N). One record is created in collections_tbl and two records are created in collection_maps_tbl.

Collections_tbl	
col_id	parent_col_id
1	null
5	null

Collection_maps_tbl		
col_id	m_id	l_id
1	M1	L1
1	M1	L2
1	M2	L1
1	M2	L2
5	M2	L1
5	M2	L2

- Non-Default Multi-Chain Implementation (chain_flag = Y). Some records are created in collections_tbl (one parent record and as many child records as there are distinct locations). Four records are created in collection_maps_tbl. Child maps are created for all items associated with the merchandise.

Collections_tbl	
col_id	parent_col_id
1	null
2 (for L1)	1
3 (for L2)	1
4 (for L3)	1
5	null
6 (for L1)	5
7 (for L2)	5
8 (for L3)	5

Collection_maps_tbl		
col_id	m_id	l_id
2	M1	L1
2	M2	L1
3	M1	L2
3	M2	L2
4	M2	L3
6	M2	L1
7	M2	L2
8	M2	L3

Scenario 6: Removing a front end pricing group and un-assigning its items.

- Before the Load: A pricing group is completely removed from the application UI. As a result, the p4p_collection record is deleted. The ITEM_DATA table is updated so that all the collection_ids are set to NULL for items belonging to the pricing group that was removed.

Collections_tbl
col_id
1
2

ITEM_DATA Table		
col_id	m_id	l_id
1	M1	L1
1	M1	L2
1	M2	L1
1	M2	L2
1	M3	L4
2	M4	L6

Items	
m_id	l_id
M1	L1
M1	L2
M2	L1
M2	L2
M2	L3
M3	L4
M4	L6
M4	L7

- After the Load:
  - Default Single-Chain Implementation (chain_flag = N). All corresponding ITEM_DATA table pricing group maps are deleted. If a map exists that contains merchandise that is not in the ITEM_DATA table, then the collection_tbl record is not removed. If no map exists, then the collection_tbl record is deleted.

Collections_tbl	
col_id	parent_col_id
1	null
2	null

Collection_maps_tbl		
col_id	m_id	l_id
1	M1	L1
1	M1	L2
1	M2	L1

Collection_maps_tbl		
1	M2	L2
1	M3	L4
2	M4	L6
2	M4	L7

- Non-Default Multi-Chain Implementation (chain_flag = Y). All corresponding ITEM_DATA table pricing group maps are deleted. If a map exists that contains merchandise that is not in the ITEM_DATA table, then the collection_tbl parent and child record are not removed.

Collections_tbl	
col_id	parent_col_id
1	null
2 (for L1)	1
3 (for L2)	1
4 (for L3)	1
5 (for L4)	1
2	null
6 (for L6)	2
7 (for L7)	2

Collection_maps_tbl		
col_id	m_id	l_id
2	M1	L1
2	M2	L1
3	M1	L2
3	M2	L2
4	M2	L3
5	M3	L4
6	M4	L6
7	M4	L7

9

Flexible Store Clustering

Flexible Store Clustering is used to customize store grouping and is implemented in Markdown Optimization using the Standard Interface and the Standard Load.

This chapter contains the following sections:

Introduction

Flexible Store Clustering is an optional feature of Markdown Optimization that permits retailers to group stores differently for different sets of merchandise. Grouping the stores in this way can facilitate more accurate optimizations and forecasts than can occur at the chain level, because the selling patterns for the set of merchandise in the stores of a given cluster will be similar.

The design of a retailer's flexible store clustering is part of the analytical configuration.

Technical Details

The following are technical details that should be taken into consideration when implementing Flexible Store Clustering:

A cluster is an arbitrary grouping of physical store locations. A cluster set is a group of clusters that is assigned to an entry in the merchandise hierarchy. A cluster set contains all stores only once.

The ideal number of groupings or clusters may vary by merchandise and retailer from approximately 5 to 25 for each set of merchandise.

Flexible Store Clustering is enabled and disabled by an implementation-wide flag that is stored in the database. The value is loaded using ASH_CP_TBL (intersect_name = CLUSTER - the new hierarchy TYPE). The intersect name of the flag is CLUSTER, with merchandise and location values set to values other than CHAIN and CHAIN. These values must match the entries in the client_hierarchy_levels_tbl. for the merchandise hierarchy and original location hierarchy levels.

If the merchandise levels are:

- Chain
- Company
- Division
- Department

- Class
- SKU

and the location levels are:

- Chain
- Region
- Store

and the cluster levels are:

- Chain
- Cluster Set
- Cluster
- Store

then the entries in ASH_CP_TBL for the CLUSTER entry must come from these defined hierarchy levels. In this example, there is only one valid location (level 2 - Cluster Set) and five valid merchandise possibilities. In general, the cluster set mappings should be set at the Division level.

Flexible Store Clustering is associated with only one level of the merchandise hierarchy for an implementation.

When merchandise hierarchy and location hierarchy values are set at a valid level other than CHAIN, clustering is enabled. If flexible store clustering is enabled, all items and worksheets are defined using flexible store clustering and all activity aggregations use the flexible store clustering aggregations.

When flexible store clustering is being used, clusters (not cluster sets) are the location optimization level.

If User Management and the Business Rule Property Manager are configured before Flexible Store Clustering is implemented, then the rules must be re-defined for any setting below the level defined for clustering.

Business rules and security can be defined at the chain, cluster, and cluster set level.

Historic data can be re-aggregated after clusters are defined by re-loading the historic data. If store clusters are reorganized, historical data must be re-loaded so that it can be re-aggregated into the new store clusters.

The values in the ASH_CP_TBL identify whether Flexible Store Clustering is being used or not. If it is being used, the location load procedure combines the location hierarchy information with the store clusters and cluster sets and populates the location hierarchy tables with the appropriate data values. The current location hierarchy information is stored in a separate table. For information on standard load validations for Flexible Store Clustering, see the *Oracle Retail Markdown Optimization Operations Guide*.

During the item creation process the items are defined as the cross product of a Merchandise Hierarchy entry and a store cluster. Worksheets should be defined at or above the cluster set level in the Location Hierarchy.

Modification of clusters to handle new location entries must be done after the locations are entered into the application. If merchandise is added to the merchandise hierarchy at a level that does not have a cluster set defined, then no cluster set will be assigned to the merchandise.

Intermediate levels between clusters and cluster sets are permitted in the Standard Load. For a CLUSTER implementation alone, this acyclic tree validation is turned OFF for the last level. However, the CLUSTER level should be the (*n*-1), where *n* is the deepest level of the hierarchy (STORE). Only the CLUSTERSET is mapped to a merchandise level and is the same for the cluster mapping interface from the retailer. The CLUSTER key (client_load_id) cannot be repeated between cluster sets.

Moving a store from one cluster to another does not translate to items re-allocation on the activities history reconciliation.

Flexible Store Clustering does not require any configuration of nor is there any impact on inference rules.

The Mhrename standard interface does not remove inactive cluster sets. So, items that are members of inactive cluster sets are filtered out during the population of the INTERNAL_ITEM_DATA_TBL table in FELOAD in load_statements.sql.

10

Understanding the Application (GUI) Configuration

This chapter introduces concepts that will help you to plan the configuration of the MDO application UI.

This chapter contains the following sections:

Introduction

This chapter provides the conceptual background necessary for you to understand how to configure the application UI to reflect the retailer's business rules and guidelines.

This chapter covers the following:

- Retailer business requirements
- The application configuration files
- The screen configuration

Note: The Grid Designer is a Web-based rich application that enables you to modify the following main components of the grid configuration in a graphical user interface:

- Column definitions
- Grid resources
- Grid configuration files
- Report configuration files

For more information on the Grid Designer, refer to the *Oracle Retail Markdown Optimization Grid Designer User Guide.*

Retailer Business Requirements

Business rules are operational constraints and guidelines that the application uses when making pricing and markdown decisions. These rules, which vary from retailer to retailer, determine the layout and functioning of the application.

Business rules determine such factors as:

- required data for optimization operations
- data display in the user interface, including layout and contents
- constraints for marking down items or pricing groups
- rules for calculating particular values

Because documented retailer business requirements are essential for the application configuration, these requirements must be gathered before you begin your configuration tasks.

The following table shows typical Markdown Optimization business requirements.

Table 10–1 Typical Markdown Optimization Business Requirements

Category	Business Requirement
Merchandise display hierarchy	Level for creating worksheet
	Level for processing markdowns
Display information criteria	Summary metrics
	What If
	Sorting and filtering
	Number and value of price ladders
	Formulas
Text strings	Screen header
	Row and column headings
	Buttons
	Drop-down menus
	Diagnostic messages
Layout	Needed grids
	Column configuration
	Column display
	Column content
	Item details
User related	Default view label and structure
	User administrative data

You configure the application interface based on the retailer's business requirements.

The Markdown Optimization Application Configuration Files

Retailers use Markdown Optimization to view selected markdown data that is derived from application's database tables.

To configure the application, you need to modify various files belonging to the application configuration set. These files are located in the configroot/p4pgui/ directory. These files consist of application-level files, column files, grid files, data definition files, and user message files.

You must reconfigure the configuration files for each new retailer installation. While some properties for a particular installation may have the same values as the default values, you may need to customize other values.

Application-Level Configuration Files

Application-level configuration files contain information specifying elements in more than one screen in the application.

These files specify elements that are not specific to columns or grids, for example, user messages, menu labels, or the time display.

Table 10–2 Application-Level Configuration Files

File Name	Defines
config.properties	The main application properties file that contains a list of all XML files that must be loaded when the application starts up. This file shows the application where to find the needed column, grid, and properties files.
p4pgui-config.xml	Defines various elements not defined in other configuration files, for example, grid names, metric item properties, user administration settings, and what-if properties. Note that the valid elements for this file are defined in the following table.
p4pguiResources.properties	Define application properties such as markdowns.
UserMessagesResources.properties	A user message that is triggered by an event within the application.
formats.properties	Defines custom formats such as price, date, percent, number, location.
CommonMessages.properties	Error message strings, command names, and minor formatting information for numeric and date columns.
p4pguiResources.properties	Error message strings and formatting properties.

The following table shows the valid elements and sub-elements (nested elements) for the p4pgui-config.xml file.

Table 10–3 Valid Elements and Nested Elements for p4pgui-config.xml File

Element	Element Description	Valid Attributes for Element
Retailer	Miscellaneous configuration settings.	name hierarchy-levels-above-worksheet last-displayed-hierarchy-level hierarchy-price ladder item-worksheet-title feedback-email feedback-dev-email feedback-mailhost report-email temp-markdowns-active(true or false) max-worklist-query-size excel-export-leaf-nodes-only collectionCentricOTB (true or false)
forecast-params	Miscellaneous attributes for the What If and Recommended Forecast screens.	show-weeks extended-summary edit-before-effective-date decreasing -prices number-forecast-weeks label-position-in-week forecast-current-week
forecast-view-row	Specifies which rows display on the Recommended Forecast screen.	NA
hierarchy	Specifies data sources for the hierarchy filter widgets.	html-form-name id key
items-metric	Specifies the layout of the summary metrics area at the bottom of worksheets.	NA
merchandise-maint-params (Nested elements: outdate-constraints, excluded-days)	Specifies the following: ■ Valid outdate range ■ Whether users modify Target Sell thru or Ending Inventory Target	NA
metrics (Nested elements: metric-m1budget, metric-m2budget, metric-fixed, metric-items)	Defines which summary worksheet-level metrics to compute, for example: ■ Name ■ Column reference ■ Aggregation type	NA
metrics-params	Specifies the display of the effective date on worksheets.	NA
page	Specifies which grids are available on the Worksheet and Maintaining Merchandise screens.	NA

Table 10–3 (Cont.) Valid Elements and Nested Elements for p4pgui-config.xml File

Element	Element Description	Valid Attributes for Element
sendback (Nested elements: select-query, pre-sendback-update)	Defines database queries that report changes made to the application metrics.	sendback name
what-if-view-column	Assigns labels and descriptions to the summary columns on the left side of the What If screen.	NA
what-if-view-row	Specifies which rows display on the What If screen.	NA
worksheet-params	Does the following: ■ Specifies column on which to search in the Find dialog box ■ Toggles the availability of the accelerated markdowns functionality.	NA

Markdown Optimization Column Configuration Files

The column configuration files consist of two XML files and one properties file, as shown in the following table.

Table 10–4 Markdown Optimization Column Configuration Files

File Type	File Name	Purpose
XML	p4p-column-list.xml	All columns available to all grids in a given retailer installation. This file is pre-configured and comes with the standard application.
	p4p-custom-columns.xml	Retailer-specific column definitions.
Properties	gridResources.properties	A file containing the column label text and description. An alias in the p4p-column-list.xml or p4p-custom-columns.xml file maps to the corresponding label text and description contained in this file.

The XML column configuration files, p4p-column-list.xml and p4p-custom-columns.xml, provide a set of fields (columns) that are available in the application across all grids. They define two types of data:

- Visible data that is displayed to the retailer in the user interface.
- Internal-use fields that are used by the system and cannot be seen by the retailer, for example, ID fields such as item ID and location ID.

These files are virtually identical in structure and syntax. The only syntax difference is in the file key name, shown in the following table. To see examples of the use of these columns in the XML column files, see the excerpts from the p4p-column-list.xml file and the p4p-custom-columns.xml file.

Table 10–5 File Key Names for XML Column Files

File Name	File Key Name
p4p-column-list.xml	internalColumns
p4p-custom-columns.xml	customColumns

These two XML column files differ mainly in the source of their column definitions.

- The p4p-column-list.xml file contains standard column definitions for the installation that come out of the box with the application.

  These column definitions are listed in the Standard Column List.xls.

- The p4p-custom-columns.xml file is where you define custom columns for a particular retailer installation.

  Typically, retailers request custom columns, which are then specified in the application metrics (business requirements) spreadsheet. These specifications define the column's display features, data source and type, and other attributes.

  This file is optional. You should use it only if the retailer installation requires custom column definitions.

  Note that the definitions in p4p-custom-columns.xml override the definitions in p4p-column-list.xml.

The columns defined in the XML column files work through the principle of inheritance. All columns defined in the p4p-column-list.xml and p4p-custom-columns.xml files are parent columns. Specifying inheritance is discussed in the section on the application grid files.

The syntax for the p4p-column-list.xml and p4p-custom-columns.xml files is the same except for the file key name.

Following is a table showing all XML elements found in both the p4p-column-list.xml file and the p4p-custom-columns.xml file. This table shows the level in the hierarchy for these elements and the acceptable values for those elements containing properties.

Table 10–6 Elements in XML Column Files

Level	Element	Purpose	Acceptable Values
Top	<column-list>		NA
Child	<column-def>	Default description of grid column.	NA
NA	<key>	The reference name for this column.	A unique name that describes the column. When creating this name, do not use spaces or HTML special characters.
NA	<column-def-properties>	Properties that define the data and display details of a <column-def>.	NA

The <column-def-properties> element accepts the following values:

Table 10–7 Acceptable Values for <column-def-properties> Element in the XML Column Files

Attribute	Value	Description
<key>	Alphanumeric characters excluding spaces and HTML special characters	A unique key name for the column that must match the key name defined in the grid file for the column.
label	Must be a valid Java property key	Alias that points to the comparable label in the gridResources.properties file containing the label text that displays at the top of the column.

Table 10–7 (Cont.) Acceptable Values for <column-def-properties> Element in the XML Column Files

Attribute	Value	Description
description	Must be a valid Java property key	Alias that points to the comparable label description in the gridResources.properties file containing the context-sensitive label description for the top of the column.
type	currency date double integer number percent string	The type of data that can appear in this column, such as text, date, number, percent. This data type must match the data type of the comparable field in the database table from which it is drawn.
display-type	blank button checkbox combobox date dropdown edit float hyperlink integer lock owner-drawn pic picture static-text time	How to render this column data on the screen.

Table 10–7 (Cont.) Acceptable Values for <column-def-properties> Element in the XML Column Files

Attribute	Value	Description
read-only-type	blank button checkbox combobox date dropdown edit float hyperlink integer lock owner-drawn pic picture static-text time	When a grid is in a read-only state, use this display type instead of the value of the display-type attribute.
DB-table-name		Defines the name of the database table that serves as the data source for this column.
DB-column-name		Defines the name of the column within the database table that serves as the data source for this column. Note that the value of the db-column-name property must be UPPER CASE.
composeable	true false	Indicates columns that the retailer can use to create custom columns.
filterable	true false	Indicates whether the retailer can choose to not display this column.
sortable	true false	Allows the grid's rows to be sorted by the column.
orderable	true false	Allows the column to be reordered in the grid relative to the columns around it.
hideable	true false	Indicates whether the user can hide the column.
expandable	true false	Indicates whether the user can expand the column.
visibility	never visible not visible visible	Specifies the visibility of the column. "never visible" means that the column is used by XML without being visible to the user through screens and drop-downs, while "not visible" means that the column is not visible.
editable	true false	Indicates whether the user can make edits to the column.

Table 10–7 (Cont.) Acceptable Values for <column-def-properties> Element in the XML Column Files

Attribute	Value	Description
filtertype	date dropdown text text area	Specifies the type of user-entry widget to use for each filterable column in the Customize Table user interface.
operatortype	equals list numeric	Specifies operator list types that are available for each filterable column in the Customize Table user interface.
columntype	none expand-collapse row-selectspacer	Specifies column types to be treated as a group. Typically, you do not modify these. The default value, none, is appropriate for any column that you customize.
function	NA	Defines functions and arguments. Note that the acceptable values for these functions are described in the following table.

The following table shows the functions that are available for the function attribute in the <column-def-properties> element.

Table 10–8 Available Functions for <column-def-properties> Element in the XML Column Files

Function	Description	Num	Date	String	Args/Type
P4P_SUM	Sum of child rows.	X			
P4P_MAX	Maximum of child rows.	X	X		
P4P_MIN	Minimum of child rows.	X	X		
P4P_AVG_CHILD	Average of child rows.	X			
P4P_AVG	Weighted average of child rows. Note that this function requires a column key as an argument, using the XML tag <args>.	X			<column-def> or <column> key
P4P_PRICELADDER	Generates price ladders.	X			
P4P_LADDERPICKER	Generates a drop-down list from which the user selects ladders.	X			

Table 10–8 (Cont.) Available Functions for <column-def-properties> Element in the XML Column Files

Function	Description	Num	Date	String	Args/Type
P4P_SUBSTITUTE	Substitutes this column with the maximum value of the child rows of the column passed as an argument.	X			<column-def > or <column> key
P4P_BLANK	Specifies that no data is generated or displayed for this column.	X	X	X	
P4P_TEMPLATE	Substitutes the data for this column with the argument.	X	X	X	Resource Key
P4P_SAME_OR_NULL	Displays a value only if all children are the same.	X	X	X	
P4P_IF_POSITIVE	Displays the value passed as the argument only if the maximum of all the children is greater than 0.	X			<column-def > or <column> key
P4P_SAME_OR_TEMPLATE	Displays vlaue of identical children else display template value.	X	X	X	Resource Key
P4P_MAP_STATUS_AND_ SAME_OR_TEMPLATE	Same as P4P_ SAME_OR_ TEMPLATE, except considers resourced values rather than key value.			X	Resource Key
P4P_TEMPLATE_IF_NOT_NULL	Show template value unless all children are null.	X	X	X	Resource Key
P4P_LADDERGROUPPICKER	Deprecated. Only available for backward compatibility with older versions of MDO.				
P4P_CHILDREN_CHECKED	Deprecated. Only available for backward compatibility with older versions of MDO.				

Data Sources in XML Column Files

The metrics that display on the application screens are either directly quoted from metrics in the database tables or are derived from calculations on metrics that come directly or indirectly from the database tables.

- Direct Metric. This type of metric is a direct reference to a column that exists as a field in the corresponding database table. A direct metric is displayed on the screen exactly as it is defined in the database.
- Derived metrics. Derived metrics are based on calculations that are made on other metrics. The two types of derived metric are:
  - Simple derivation. This type of metric is derived from the result of a formula that performs calculations on a metric that comes directly from a database column.
  - Complex derivation. This type of metric is derived from a column that is defined in one of the XML column files, which in turn is derived from a column in the database. That is, an XML column may refer to another XML column that directly refers to the database.

Sorting on User-Defined Metrics

The p4p-column-list.xml file can be modified to provide the user with the ability to sort on user-defined metrics. To do this:

1. In the XML file, go to USER_DEFINED_SUM, USER_DEFINED_DIFF, USER_DEFINED_RATIO, and USER_DEFINED_CUM_SUM columns.
2. Change the sortable flag from sortable=false to sortable=true.

String columns in p4p-column-list.xml can be configured to make filtering case sensitive or case insensitive. To do this, use the following tags:

- iscasesensitive=false to make the column case insensitive
- iscasesensitive=true to make the column case sensitive

Hierarchy Filtering

By default, a user can filter by Hierarchy 6. To use another level, change the INT_UNIQUE_ID derivation in p4p-column-list.xml to the appropriate number.

A user can filter items by any unique hierarchy combination. For example, configure INT_UNIQUE_ID to HIERARCHY(*n*)-HIERARCHY(*n*+1)-REGION in order to filter items by the HIERARCHY(*n*)-HIERARCHY(*n*+1)-REGION hierarchy combination.

The gridResources.properties file contains the label and description for the column.

Margin Visibility

The following applies to the Margin Visibility feature of What If.

p4p-column-list.xml

Changes to p4p-column-list.xml

Column Groupings

The following column groupings have been added for MDO13.1

```
<!-- COLUMN GROUPS -->
   <column-def>
```

```
          <key>MODEL_RUN_METRICS_COLUMN_GROUP</key>
          <column-def-properties groupId="GROUP_HEADER"
label="p4pgui.modelrun.metrics.group.column.label" group-
description="p4pgui.modelrun.metrics.group.column.label" orderable="true"
hideable="true" sortable="false"/>
    </column-def>
     <column-def>
          <key>CURRENT_METRICS_COLUMN_GROUP</key>
          <column-def-properties groupId="GROUP_HEADER"
label="p4pgui.cur.metrics.group.column.label" group-
description="p4pgui.cur.metrics.group.column.label" orderable="true"
hideable="true" sortable="false"/>
    </column-def>
<!-- End Column Groups -->
```

Columns for Current Metrics (Items)

Internal Grid Name	Direct From DB or Grid Calculated	Source Table/View	Sourcing Details
curNextRecMDDate	Direct	p4p_display_items	CUR_PROJECTED_NEXT_ MARKDOWN
curRecMDDollarsCost	Grid	p4p_display_items	(CASE WHEN PROCESS_AS_ITEM = 1 THEN RECOMMENDED_ITEM_FLAG ELSE RECOMMENDED_ COLLECTION_FLAG END)*(NVL (CUR_PROJ_OH_UNITS_EFF_DT, 0) * UNIT_COST)
curTakenMDDollarsCost	Grid	p4p_display_items	CASE MARKDOWN_FLAG WHEN 0 THEN 0 ELSE NVL (CUR_PROJ_OH_ UNITS_EFF_DT, 0) * UNIT_COST END
curProjOutDate	Direct	p4p_display_items	CUR_PROJECTED_OUT_OF_STOCK
curProjOnHandUnitsEffDt	Direct	p4p_display_items	CUR_PROJ_OH_UNITS_EFF_DT
curProjSalesDollarsEOL	Grid	p4p_display_items	CUR_EOL_CUM_DOLLARS_SALES + CUMULATIVE_SALES_DOLLARS
curProjLifeRetail	Grid	p4p_display_items	CUMULATIVE_SALES_DOLLARS + CUR_EOL_CUM_DOLLARS_SALES
curProjSalesUnitsEOL	Grid	p4p_display_items	CUR_EOL_CUM_UNIT_SALES + CUMULATIVE_QUANTITY_SOLD
curProjSellThruEOL	Grid	p4p_display_items	(CASE NVL (CUMULATIVE_ QUANTITY_SOLD,0)+ NVL (CUR_ EOL_CUM_UNIT_SALES,0)+ NVL (CUR_ENDING_INVENTORY_UNITS,0) WHEN 0 THEN 0 ELSE (1-ROUND(CUR_ENDING_ INVENTORY_UNITS/(NVL (CUMULATIVE_QUANTITY_SOLD,0)+ NVL (CUR_EOL_CUM_UNIT_ SALES,0)+ NVL (CUR_ENDING_ INVENTORY_UNITS,0)), 4)) END)
curProjSalesUnitsEOLandEIU	Grid	p4p_display_items	(NVL (CUMULATIVE_QUANTITY_ SOLD,0)+ NVL (CUR_EOL_CUM_ UNIT_SALES,0)+ NVL (CUR_ENDING_ INVENTORY_UNITS,0))
curProjOnHandUnitsEOL	Direct	p4p_display_items	CUR_ENDING_INVENTORY_UNITS

Internal Grid Name	Direct From DB or Grid Calculated	Source Table/View	Sourcing Details
curProjOnHandRtlDollarsEOL	Grid	p4p_display_items	CUR_ENDING_INVENTORY_UNITS * CUR_REC_RTL_MIN
curProjOnHandCostDollarsEOL	Grid	p4p_display_items	CUR_ENDING_INVENTORY_UNITS * UNIT_COST
curProjMarginDollarsEOL	Direct	p4p_display_items	CUR_PROJ_STD_EOL_GM_AMOUNT
curProjMarginPercEOL	Direct	p4p_display_items	CUR_PROJ_STD_EOL_GM_PERC

Columns for Current Metrics (Groups)

Internal Grid Name	Direct From DB or Grid Calculated	Source Table/View	Sourcing Details
curProjSalesDollarsEOLGrp	Grid	p4p_display_items	CUR_EOL_CUM_DOLLARS_SALES_C + CUMULATIVE_SALES_DOLLARS
curProjSalesUnitsEOLGrp	Grid	p4p_display_items	CUR_EOL_CUM_UNIT_SALES_C + CUMULATIVE_QUANTITY_SOLD
curProjSellThruEOLGrp	Grid	p4p_display_items	(CASE NVL (CUMULATIVE_ QUANTITY_SOLD,0)+ NVL (CUR_ EOL_CUM_UNIT_SALES_C,0)+ NVL (CUR_ENDING_INVENTORY_UNITS_ C,0) WHEN 0 THEN 0 ELSE (1-ROUND(CUR_ENDING_ INVENTORY_UNITS_C/(NVL (CUMULATIVE_QUANTITY_SOLD,0)+ NVL (CUR_EOL_CUM_UNIT_SALES_ C,0)+ NVL (CUR_ENDING_ INVENTORY_UNITS_C,0)), 4)) END)
curProjOnHandUnitsEOLGrp	Direct	p4p_display_items	CUR_ENDING_INVENTORY_UNITS_C
curProjOnHandRtlDollarsEOLGrp	Grid	p4p_display_items	CUR_ENDING_INVENTORY_UNITS _ C* CUR_REC_RTL_MIN
curProjOnHandCostDollarsEOLGrp	Grid	p4p_display_items	CUR_ENDING_INVENTORY_UNITS_C * UNIT_COST
curProjMarginDollarsEOLGrp	Direct	p4p_display_items	CUR_PROJ_STD_EOL_GM_AMOUNT_ C
curProjMarginPercEOLGrp	Direct	p4p_display_items	CUR_PROJ_STD_EOL_GM_PERC_C

Sample Entry for p4p-column-list.xml

```
<column-def>
    <key>curProjMarginDollarsEOL</key>
    <column-def-properties label="p4pgui.cur.projMarginDollarsEOL.column.label"
description="p4pgui.cur.projMarginDollarsEOL.column.description" db-table-
name="P4P_DISPLAY_ITEMS" db-column-name="CUR_PROJ_STD_EOL_GM_AMOUNT"
type="double" display-type="currency" filterable="true" sortable="true"
orderable="true" hideable="true" groupId="GROUP_HEADER"
format="p4pgui.bigCurency.column.format" composeable="true" editable="true">
      <function key="P4P_SUM"/>
    </column-def-properties>
  </column-def>
```

Changes to Grid XMLs

The grid XML files have been updated to use the column grouping of the projected forecast metrics.

Example

```
<column-group>
        <key>MODEL_RUN_METRICS_COLUMN_GROUP</key>
        <columngroup-properties
            resourced-label="true"
            group-description="p4pgui.modelrun.metrics.group.column.label"
            resource="false"/>
        <column>
          <key>projSalesDollarsEOL</key>
          <column-properties/>
        </column>
        <column>
          <key>projMarginDollarsEOL</key>
          <column-properties/>
        </column>
        <column>
          <key>projMarginDollarsEOLGrp</key>
          <column-properties/>
        </column>
        <column>
          <key>projOutDate</key>
          <column-properties/>
        </column>
        <column>
          <key>projOnHandUnitsEOL</key>
          <column-properties/>
        </column>
      </column-group>
      <column-group>
        <key>CUR_METRICS_COLUMN_GROUP</key>
        <columngroup-properties
            resourced-label="true"
            group-description="p4pgui.current.metrics.group.column.label"
            resource="false"/>
        <column>
          <key>curProjSalesDollarsEOL</key>
          <column-properties/>
        </column>
        <column>
          <key>curProjMarginDollarsEOL</key>
          <column-properties/>
        </column>
        <column>
          <key>curProjMarginDollarsEOLGrp</key>
          <column-properties/>
        </column>
        <column>
          <key>curProjOutDate</key>
          <column-properties/>
        </column>
        <column>
          <key>curProjOnHandUnitsEOL</key>
          <column-properties/>
        </column>
 </column-group>
```

Markdown Optimization XML Grid Configuration Files

A grid is a spreadsheet-like table that defines how columns and rows display on an application screen.

Each XML grid file determines the configuration of the associated screen, which has the same name as the file. For example, an XML configuration file named worksheet-summary-grid.xml determines the configuration of the associated Worksheet Summaries screen.

The following table shows the standard set of XML grid configuration files.These files are located in the directory configroot/resources/p4pgui/grids.

Table 10–9 Standard Set of Markdown Optimization Grid Configuration Files

XML File Name	Screen Elements Defined in File
p4p-edit-group-grid.xml	Elements for adding and removing items from the pricing groups grid.
p4p-edit-items-wksht.xml	Elements for adding and removing items from the Worksheet grid.
p4p-maint-grid-groups.xml	Elements for maintaining pricing groups.
p4p-items-grid-flat.xml	Elements that open the items worksheet.
p4p-price-groups-grid.xml	Elements that open the items worksheet.
p4p-maint-grid-flat.xml	The element that opens the lowest aggregation of the Maintaining Merchandise worksheet.
p4p-promo-details-grid.xml	Define details about promotions s for a particular item.
p4p-wksht-summary-grid.xml	Elements located in the Markdown Worksheet Summaries worksheet.

Inheritance in Grid Configuration Files

Typically, worksheets are designed so that some rows display summarized data from other rows. For example, a worksheet might contain a set of adjoining rows that display the data for several different colors of the same item, with each row displaying the data for a different color. To display the aggregated data from all the different item colors, a summary row is used, thereby creating a hierarchy of rows. Thus, each row represents either a record in the database or a defined aggregation of records.

The hierarchies of rows are defined within the <row-group> element of the XML grid file. The hierarchies are specified by defining the summary data row (defined aggregation of database records) as a parent row and the item-level data (individual database records) as child-level rows. Child rows are nested within the next highest level of row, which may be either the parent row or a higher-level child row.

Note that the <row-group> element is required when configuring hierarchies of rows for an application screen. Otherwise, it is not used.

Grid File Elements and Attributes

The first XML element in a grid file is the <grid> element. This element contains:

- a <key> attribute that provides a unique reference name
- properties defining areas of the grid other than the columns

The following table shows the elements and attributes of grid files in the order in which they appear in the file.

Table 10–10 Elements and Attributes in XML Grid Files

Level	Element or Attribute	Purpose
Top	<grid>	Highest-level (root) XML element for configuring the XML grid file.
	<grid-properties>	Properties that define the grid as a whole, not the rows or columns.
Child	<column-group-spec>	Required element that specifies the columns to be displayed in the grid. It must contain <column> child elements to specify the columns. If the screen's rows are organized hierarchically, the <column-group-spec> element must also contain a <row-group> element.The column definitions are nested within this element.
	<column>	Element that defines each column to be included in the grid. It must contain a column key that maps to a column key in one of the XML column files. It may also contain column properties.
	<key>	Required attribute that specifies the column key, which is a unique identifier for a specific column that is included in the grid. This key points to the column in one of the XML column files that has the identical key.
Parent	<row-group>	Optional element that enables the display of multiple levels (hierarchical groups) of rows. It defines the highest row level of a hierarchically organized set of rows.
Child	<row-group>	Nested child row group of parent row group.
Child of child	<row-group>	Nested row group. Child of above child row group.

Note that Row group definitions override column group definitions.

Within the <row-group> element, the <groupby> element refers to a value that is defined in the column-list file. For example, the meaning of <groupby>WKSHT_HIERARCHY2</groupby> is that there should be one parent row per set of child rows whose value for WKSHT_HIERARCHY2 (as defined in a column file) is the same as the value for WKSHT_HIERARCHY2 for the parent row.

The <grid-properties> define the specific properties for each grid, independent of the rows or columns. For example, the frozenColumns property specifies how many columns do not scroll out of view during horizontal scrolling. This property defines the grid as a whole rather than a particular column.

Grid properties include the following:

Table 10–11 Grid Properties

Attribute	Example
columnsFrozen	columnsFrozen="5"
db-key-column-name	db-key-column-name="entercolumnname"
db-table-name	db-table-name="ITEM_DATA"
defaultrowlevel	defaultrowlevel="1"
filterable	filterable="false"
firstrowheadercolumn	firstrowheadercolumn="2"
readonly	readonly="true"

Table 10–11 (Cont.) Grid Properties

Attribute	Example
rowsFrozen	rowsFrozen=”2 ”
grid name	name=”p4pgui.maintGridFlat.grid.label”

Data Definition Files

These files, which consist of one schema (XSD) file and two document type definition (DTD) files, define the syntax rules for the associated XML files.

These files are located at: configroot/p4pgui/[retailer directory]/grids/

The content of each XML file in the configuration set must conform to the schema or DTD that defines it.

Table 10–12 Data Definition Files for Markdown Optimization XML Files

File Name	Purpose
grid.xsd	The XML schema that defines all but two of the application XML configuration files.
p4pgui-config.dtd	The DTD that defines the p4pgui-config.xml file.
item-detail-layout.dtd	The DTD that defines the item-details-layout.xml file.

The Markdown Optimization Screens

Retailers view and work with their markdown data using a series of application screens. These screens may be classified according to a retailer-based perspective or a configuration-based perspective.

Retailer-Centered Classification of Screens

These screens display grids, which are two-dimensional data structures composed of rows and columns; they are similar to a spreadsheet. All application screen grids are based on a common grid infrastructure.You can organize the rows and columns on the grid to display hierarchically to as many levels of aggregation as the retailer wants.

From the retailer’s viewpoint, the two types of screens are:

- Worksheet screens
- Reports

Worksheet Screens Each worksheet screen displays a grid showing a grouping of items, with associated pricing information. Retailers use worksheets to:

- view item details
- change markdown prices
- accept or decline markdown recommendations
- select items for forecasting and what-if analysis
- maintain pricing groups, for example, add or remove items from pricing groups
- select items for forecasting, what-if analysis and optimize-to-budget
- view results of pricing decisions
- save changes

Each worksheet represents a department or level in the merchandise hierarchy, such as chain, zone, or pricing group. Retailers can access only those worksheets for which the retailer's system administrator has given permissions.

The application worksheet screens are shown in the following table:

Table 10–13 Markdown Optimization Worksheet Screens

Screen	Purpose
Worksheets Summaries	Displays a list of the worksheets that the retailer can access, along with a summary of each worksheet and its current status.
(Item) Worksheet	
Edit Item	Editing screen associated with Item Worksheet that enables retailer to edit the Item Worksheet, for example, add or remove items or pricing groups that are not currently recommended for a markdown.
Pricing Groups	Screen associated with a pricing group, which is a set of items that must all be marked down together on the same day.
Edit Pricing Group	Editing screen associated with the Pricing Groups worksheet that enables the retailer to add or remove items from a pricing group.
Maintaining Merchandise	Enables the retailer to perform tasks on items and pricing groups, for example, setting and removing exit dates, setting exit inventory or sell-through targets, and creating, modifying, and removing pricing groups.

Reports Reports enable retailers to view their data in non-standard ways that are unavailable on the worksheet screens. For example, a retailer may want to see data that is usually displayed on one worksheet displayed in a different format — across multiple screens. Like worksheets, reports are displayed through an ActiveX grid. The number and type of reports that you configure are determined by the retailer's business requirements and thus vary from one installation to another.

Note that for both worksheets and reports, you cannot configure the overall design of the grid; you can only configure its rows and columns.

Configuration-Centered Classification of Markdown Optimization Screens

The three configuration types of Markdown Optimization screens are:

- Total
- Limited
- Display Only

A summary of all screens, categorized by configuration type, is shown in the following table:

Table 10–14 Markdown Optimization Screen Configuration Types

Configuration Type	Elements to Configure	Related Screens
Total	Column metrics	Worksheet summaries
	Filter and sort options	Worksheet
	Aggregation	Maintaining Merchandise
		Reports

Table 10–14 (Cont.) Markdown Optimization Screen Configuration Types

Configuration Type	Elements to Configure	Related Screens
Limited	Row and column selections	What If
		Recommended Forecast
		User Administration
		Edit User
Display only	Static text updates	Item Information
		Promo Details

Markdown Optimization Total Configuration Screens

This screen type allows you to completely configure the rows and columns on the grid, including creating new rows and columns.

The configuration of these screens defines:

- column metrics
- filtering and sorting options
- aggregations

The total configuration type screens and their functions are shown in the following table.

Table 10–15 Markdown Optimization Total Configuration Type Screens

Screen(s)	Function
Worksheet summaries	Summary screen that displays the following: ■ A list of the item worksheets that the retailer can access ■ A summary of each worksheet and its current status
(Item) Worksheets	Main screen used by retailer for pricing (markdown) decisions. Main functions are: ■ evaluating recommendations for markdowns ■ accepting or declining markdown recommendations
Edit item worksheet	Editing screen associated with Item Worksheet. Enables the retailer to edit the Item Worksheet, for example, adding or removing items or pricing groups that are not currently recommended for a markdown.
Maintaining merchandise	A page used for the following tasks: ■ Item-level tasks such as setting or removing exit dates and setting exit inventory or sell-through targets. ■ Pricing group-level tasks such as creating and deleting pricing groups and managing those pricing groups by adding and removing items and changing pricing group names.
Pricing Groups	Screen associated with a pricing group, which is a set of items that must all be marked down together on the same day.
Edit pricing groups	Editing screen associated with the Pricing groups worksheet that enables the retailer to add or remove items from a pricing group.

Table 10–15 (Cont.) Markdown Optimization Total Configuration Type Screens

Screen(s)	Function
Reports such as Markdown Analysis and Sample Price Change	Reports enable retailers to view application data in alternative ways that are not available in the worksheets. Markdown Analysis provides information to enable the retailer to assess how markdowns are being implemented throughout departments. It includes the following: ■ Items. ■ Pricing information such as original current ticketed, current ticketed or recommended price. ■ Inventory information such as on hand or on order. ■ Financial summary metrics.

Markdown Optimization Limited Configuration Screens

Limited Configuration screens provide a predefined selection of rows, columns, and other properties to use for their configuration. You can select only from these predefined rows and columns, which are specified in the p4pgui-config.xml file, located in the grids folder.

The following screens are limited configuration screens:

- What If
- Recommended Forecast
- User administration
- Edit user

Configuring the What If Screen

The What If screen enables retailers to experiment with the outcomes of various markdown scenarios by creating their own markdown schedules. Retailers can try out various non-standard outcomes that may do any or all of the following:

- Disregard the markdown recommendations
- Violate the company's business rules
- Select prices from alternative price ladders
- Defer markdown actions

The What If Screen

The What If screen modifies and displays the optimization results from that week's batch run. It displays only those items already optimized in the batch run, including all items with recommended markdowns. This screen neither re-optimizes nor performs a complete forecast.

When the retailer enters price changes into the What If screen, the application computes the sales impacts of these changes and then uses this sales and price information to compute the screen metrics. After creating a what-if scenario, the retailer typically compares the metrics calculated in this scenario to the metrics for the standard Markdown Optimization markdown recommendations.

Note that the What If screen does not apply business rules as do the other screens. Because this screen does not use the Calculation Engine, the retailer cannot see the

effects of promotions and of many business rules. Additionally, the What If code does not apply an inventory effect (also known as a pigeonholing effect) as does the Calculation Engine. Therefore, the sales forecasts that it produces may be highly speculative.

Data Sources for the What If Screen

The calculations performed on the What If page are based on data from the following sources:

- Retailer selections from the What If screen's user interface, for example, selecting price ladders from drop-down widgets such as the price ladder selection list.
- Markdown Optimization database views (which are described in the following table).

Note that before configuring the What If screen, verify that the database tables are populated with forecast data calculated by the model run.

Table 10–16 Database Data Sources for What If Screen

Database View (P4P_ ...)	Database Table (P4P_ ...)	Type of Data	Additional Information
FORECAST_DATA	FORECAST_ ACTIVITIES	Forecast data such as: • Week-by-week set of prices and sales • Base demand information	This view must be populated with forecast data calculated from the model run. The Calculation Engine calculates this forecast data, and then stores it in the FORECAST_ ACTIVITIES database table. The FORECAST_DATA view contains one row for each week of an item's forecast. Because the What If screen takes data from the database view and not directly from the database, this screen has no dynamic interaction with the Calculation Engine.
WHAT_IF	ITEM_DATA	Forecast data such as: • Full price • Price elasticity • Inventory effect parameters	This view pulls price elasticity, inventory effect, and other parameters that were used as inputs to the Calculation Engine in the model run. If the price elasticity parameter in the P4P_ WHAT_IF view is not properly populated, the retailer cannot change the forecast by modifying the price schedule on the What If screen.
DISPLAY_ITEMS	ITEM_DATA	• Current item-level data • Season-to-date summary metrics	This view provides parameters for forecast. You must ensure that the keys listed in the next table are specified in the p4pgui-config.xml file in order to pull the required fields (records) out of the ITEM_ DATA table. These keys are the key attributes of the <column-def> tags in the XML column files. The data pulled out from ITEM_DATA is used to compute various metrics such as how much inventory, outdate, and the inventory cost.

Configuring Different Areas of the What If Screen

You can configure the various areas that comprise the What If screen. These areas are (from top to bottom):

- Display panel
- Action selection area
- Grid
- Graph

The following table describes the areas of the What If screen.

Table 10–17 Areas of the What If Screen

Screen Area	Sub-Areas	Micro-Areas	Description	Configurable
Display panel	NA	NA	Displays number of items and outdate range.	NA
Action selection	NA	NA	Enables retailer to select an action, for example, What If.	NA
Grid	Predefined rows	NA	Each predefined row represents data from a specified column from an item worksheet..	Yes
NA	Columns	Display-only	These predefined columns contain display-only metrics such as summary end of life. In the p4pgui-config.xml file, you specify which metrics, that is, which columns to display in this area.	Yes
NA	NA	Weekly recommended forecast data	These columns consist of one columns for each week's worth of forecast data. The retailer can interact with these columns by changing the markdown recommendations within them. In the p4pgui-config.xml file, you specify how many weeks' worth of data to display, that is, how many weekly columns to display by setting the number-forecast-weeks attribute of the <forecast-params> tag..	Yes
Graph	NA	NA	Display-only graph showing sales and inventory.	NA

The configuration of the columns and rows in the grid are described in the following sections.

Configuring Columns for the What If Screen

The following table shows the predefined columns that you can select for the What If screen.

Table 10–18 Predefined Columns for the What If Screen

Key	Display Name	Column Name	Description
STD	STD	Season to Date	The total dollars or units from the beginning of the season to today.
TTOOS	TTOOS	Total Till Out of Stock	The forecasted total from today until the exit date.
FCEOL	FCEOL	Forecasted End of Life	The sum of the Life to Date and Total Till Out of Stock totals. This sum represents the forecasted total from the beginning of the season to the exit date, using the application recommended markdown schedule.

Table 10–18 (Cont.) Predefined Columns for the What If Screen

Key	Display Name	Column Name	Description
EOL	EOL	End of Life	The sum of the Life to Date and Total Till Out of Stock totals.

Here are the definitions for the above columns, as found in the p4pgui-config.xml file.

```
<what-if-view-column
display name="p4pgui.whatIfCol.STD.label"
description="p4pgui.whatIfCol.STD.description"
use-as="STD"/>
<what-if-view-column
display name="p4pgui.whatIfCol.TTOOS.label"
description="p4pgui.whatIfCol.TTOOS.description"
use-as="TTOOS"/>
<what-if-view-column
display name="p4pgui.whatIfCol.FCEOL.label"
description="p4pgui.whatIfCol.FCEOL.description"
use-as="FCEOL"/>
<what-if-view-column
display name="p4pgui.whatIfCol.EOL.label"
description="p4pgui.whatIfCol.EOL.description"
use-as="EOL"/>
```

The attributes in these column definitions are described in the following table.

Table 10–19 Column Attributes for What If Screen

Tag	Description
display-name	Alias that points to the comparable label in the gridResources.properties file containing the label text that displays at the top of the column.
description	Alias that points to the comparable label description in the gridResources.properties file containing the context-sensitive label description for the top of the column.
use-as	The calculation performed to produce the displayed result.

To configure the weekly recommended forecast data columns that display on the right-hand side of the grid, you set the number-forecast-weeks attribute of the <forecast-params> tag in the p4pgui-config.xml file to the number of weeks' worth of forecasts that you want to display. One column displays in the screen for each week's forecast.

For example, the following tag is set to display 15 weeks' worth of data. This means that 15 right-hand columns display.

```
<forecast-params
number-forecast-weeks="15"/>
```

Note that the What If page does not display forecasts that go beyond the last outdate, even if you select a number of forecast weeks that is larger than the number of weeks till the last outdate.

The default configuration for the maximum number of weeks to display is 10. You can theoretically set a value as high as 104, which would allow for two years' worth (104 weeks' worth) of data to display. However, displaying more than 10 or 15 forecast weeks is unwieldy and is not recommended.

Configuring Rows for the What If Screen

The predefined rows that you can select for the What If screen are shown in the following table.

Table 10–20 Predefined Rows for the What If Screen

Key	Display Name	Pop-Up Description
fillWidgets	Fill right / left	Displays left and right arrows that the retailer can click to propagate selections to the left or right.
forecastRecommendedTicket	Rec Ticket Price	The forecasted ticket price at the end of the week if all recommended markdowns are taken.
forecastPrice	Rec Sales Price	The forecasted sales price for the week if all recommended markdowns are taken.
ladderID	Price Ladder ID	Drop-down list of price ladder names.
whatIfPrice	Override Price	The retailer-entered what-if price.
forecastTicketPrice	New Ticket Price	The what-if ticket price based on the retailer's Override Price entries.
registerPrice	New Sales Price	The what-if sales price based on the retailer's Override Price entries.
salesDollars	Sales $	The total sales dollars for the specified time period.
salesUnits	Sales Units	The total sales units for the specified time period.
gmDollarsCost	GM $ (Cost)	The gross margin derived by calculating the cost of goods as follows: sales units * unit cost.
gmPercentCost	GM % (Cost)	The gross margin percent derived by calculating the cost of goods as follows: sales units * unit cost.
gmDollarsRetail	GM $ (Retail)	The gross margin percent derived by calculating the cost of goods as follows: (sales dollars + markdown dollars) * (unit cost / original retail price).
gmPercentRetail	GM % (Retail)	The gross margin derived by calculating the cost of goods as follows: (sales dollars + markdown dollars) * (unit cost / original retail price).
markdownDollars	Markdown $	The forecasted markdown cost for the specified time period based on both permanent and temporary markdowns.
sellThroughPercentRemaining	Sell Thru %	The percentage of inventory sold through at the end of the specified time period.
inventoryUnits	EOH Units	The remaining units on hand at the end of the specified time period.
inventoryDollarsRetail	EOH $ (Cost)	The cost of the remaining units on hand.

Table 10–20 (Cont.) Predefined Rows for the What If Screen

Key	Display Name	Pop-Up Description
inventoryDollarsCost	EOH $ (Retail)	The retail value of the remaining units on hand.
promoFlag	Promo Flag	Indicator for a planned promotional event. You must configure this row to display on the What If screen if you want to create Promo hyperlinks that enable the retailer to display the Promo Details popup screen.

An example of a predefined row for the What If screen is What If Price. This definition is found in the p4pgui-config.xml file.

```
<what-if-view-row
display name="p4pgui.whatIfRow.priceLadder.label"
description="p4pgui.whatIfRow.priceLadder.description"
use-as="whatIfPrice"
type="forecast-dropdown"
format=p4pgui.whatIfRow.forecastPrice.format"/>
```

The following table describes these row attributes.

Table 10–21 Row Attributes for What If Screen

Tag	Description	Additional Information
display-name	Alias that points to the comparable label in the gridResources.properties file containing the label text that displays to the left of the row.	NA
description	Alias that points to the comparable label description in the gridResources.properties file containing the context-sensitive label description for the row.	NA

Table 10–21 (Cont.) Row Attributes for What If Screen

Tag	Description	Additional Information
use-as	The calculation performed to produce the result displayed in the row.	Acceptable values are: ladderID inventoryUnits inventoryDollars inventoryDollarsCost inventoryDollarsRetail sellThroughPercent Remaining sellThrough Percent Total gMPercentCost gMDollarsCost gMPercentRetail gMDollars Retail salesUnits sales Dollars markdownDollars forecastPriceAverage forecastPriceExact forecastPriceAveragePercent forecastPriceExactPercent promoFlag
type	Data type, for example, floating point, integer, string.	Acceptable values are: integer money percent flag
format	Alias that points to the comparable formatting information in the gridResources.properties file.	This formatting information determines the formatting of numeric data, for example, the use of commas and decimal points.

Configuring Input Columns for What If Screen Rows

The What If screen takes data from certain column definitions found in the p4p-column-list.xml file. The application uses this data to calculate the information displayed in the What If screen rows. These required column definitions, which are pre-configured as part of the application, are shown in the following table.

The following table shows the required columns for the P4P_DISPLAY_ITEMS database view. These column definitions are contained in the p4pgui-config.xml file.

Note that for items to display properly on the What If screen, you must verify that the following column keywords are correctly configured in the p4pgui-config.xml file. If an item has missing or incomplete data, the What If screen does not factor that item into its calculations and therefore may display obsolete data.

Table 10–22 Correct Configurations for What If Keywords

Column Name	Keyword	Description
NA	INT_OUT_OF_STOCK_DATE	If an item has no outdate, it does not receive a forecast (markdown recommendation) and is therefore ignored.
Internal inventory	INT_INVENTORY	What If ignores items with null or zero current inventory.
NA	INT_MOST_RECENT_RETAIL	Not needed for truncating the price ladders presented to the retailer or for correct calculation of markdown costs.
NA	INT_TICKET_PRICE	Not needed for truncating the price ladders presented to the retailer or for correct calculation of markdown costs.
Internal unit cost	INT_UNIT_COST	Determines the outcome of certain other calculations such as inventory valuation and gross margin.
Internal season-to-date average price	INT_STD_AVG_PRICE	Determines the calculation of several other season-to-date metrics such as STD sales dollars and STD GM%.
Internal cumulative sales	INT_CUM_SALES	Determines the calculation of several other season-to-date metrics such as STD sales dollars and STD GM%.

Table 10–23 Required Column Definitions in p4p-column-list.xml for What If Screen Rows

Column Name	Column Key
Internal season-to-date average price	INT_STD_AVG_PRICE
Internal cumulative sales	INT_CUM_SALES
Internal inventory	INT_INVENTORY
Internal unit cost	INT_UNIT_COST
Internal season-to-date sell-through percent	INT_STD_SELLTHRU_PERC
Internal current percent off original	INT_CURRENT_PERC_OFF_ORIG

Note that when you configure the What If screen, you must ensure that all these column definitions are specified in the p4p-column-list.xml file.

Configuring Other Features of the What If Screen

In addition to configuring the rows and columns for the What If screen, you may need to configure the following screen features.

- Forecast parameters

- Display of metrics such as:
  - Price ladder
  - Price
  - Markdown cost
  - Gross margin dollars
  - Gross margin percent
  - Sell-through percent
  - Sales dollars and inventory dollars

spreadCurrentInventoryUnits – By default, Current Inventory Units are applied to each item, but the value is spread across items for a spread based on the specified method.

Specifying the forecast parameters: The only currently supported attributes for the <forecast-params> element in the p4pgui-config.xml file are shown in the following table.

Table 10–24 Forecast Parameters Supported Attributes

Attribute	Description	Acceptable Values
number-forecast-weeks	Number of weeks of forecast data to display on the screen	Up to 104 (which enables the display of two years' worth of data)
edit-before-effective-date	Whether have option of modifying the price before the effective date	true (yes) false (no)

Note that if any other elements are listed within the <forecast-params> element, delete them because they are no longer supported.

Specifying the metrics: The following table shows the What If row metrics.

Table 10–25 What If Row Metrics

Row Metric	Description
Price ladder	Contain the drop-down selection list (widgets) in which users specify the markdown selection. Note that you must ensure that the <grid-display> attribute for this metric is set at "allowed." Otherwise, the What If screen does not function.
Price	Prices recommended by the optimizer, that is, the suggested values of the register price. Note that the ticket price may be higher than the register price because of point-of-sale (POS) markdowns.
Markdown Cost	The value of the price change multiplied by the inventory on hand at the moment the markdown takes effect. Price change is defined as the difference between the last ticket price and the current register price.
Gross Margin Dollars	Gross margin dollars calculated without regard to business rules or restrictions that define alteration costs and cash discounts. As a result, the What If screen may calculate markdowns that make the EOL gross margin larger than the recommended forecast.

Table 10–25 (Cont.) What If Row Metrics

Row Metric	Description
Sell Through Percent	Percentage of inventory sold through a specified time period.
Sales Dollars	Dollar value of merchandise sold in a period.
Inventory Dollars	Dollar value of the on-hand inventory.

The Recommended Forecast Screen

The Recommended Forecast screen displays view-only summary metrics for the weekly forecast for all items with forecast information, which is based on Markdown Optimization's markdown recommendations. Because its rows and columns are already defined, this screen requires minimal configuration.

Before configuring this screen, ensure that the ITEM_DATA table is populated with data. This table stores the following forecast-related fields:

- Forecast identifier
- Opportunity cost
- Projected end-of-life (EOL) inventory
- Projected gross margin
- Current and next week's sales
- Markdown information

Configuring Columns for the Recommended Forecast Screen

The following table shows the predefined columns that you can select for the Recommended Forecast screen.

Table 10–26 Predefined Columns for Recommended Forecast Screen

Key	Display Name	Column Name	Description
STD	STD	Season to Date	The total dollars or units from the beginning of the season to today.
TTOOS	TTOOS	Total Till Out of Stock	The forecasted total from today until the exit date.
FCEOL	FCEOL	Forecasted End of Life	The sum of the Life to Date and Total Till Out of Stock totals, representing the forecasted total from the beginning of the season to the exit date, using the application recommended markdown schedule.
EOL	EOL	The sum of the Life to Date and Total Till Out of Stock totals.	End of Life

Here are the definitions for the above columns, as found in the p4pgui-config.xml file. Note that these column definitions begin with the element tag <what-if-view-column>.

Note that all columns defined as What If columns in the p4pgui-config.xml file can also be used as Recommended Forecast columns.

```
<what-if-view-column
```

```
display name="p4pgui.whatIfCol.EOL.label"
description="p4pgui.whatIfCol.EOL.description"
use-as="EOL"/>
<what-if-view-column
display name="p4pgui.whatIfCol.TTOOS.label"
description="p4pgui.whatIfCol.TTOOS.description"
use-as="TTOOS"/>
<what-if-view-column
display name="p4pgui.whatIfCol.STD.label"
description="p4pgui.whatIfCol.STD.description"
use-as="STD"/>
<what-if-view-column
display name="p4pgui.whatIfCol.FCEOL.label"
description="p4pgui.whatIfCol.FCEOL.description"
use-as="FCEOL"/>
```

The following table describes the column attributes found in these column definitions.

Table 10–27 Column Attributes for Recommended Forecast Screen

Tag	Description
display-name	Alias that points to the comparable label in the gridResources.properties file containing the label text that displays at the top of the column.
description	Alias that points to the comparable label description in the gridResources.properties file containing the context-sensitive label description for the top of the column.
use-as	The calculation performed to produce the result displayed in the column.

Configuring Rows for the Recommended Forecast Screen

The following table shows the predefined rows for the Recommended Forecast screen.

Table 10–28 Predefined Rows for the Recommended Forecast Screen

Key	Display Name	Pop-Up Description
salesDollars	Sales $	The total sales dollars for the specified time period.
salesUnits	Sales Units	The total sales units for the specified time period.
gmDollarsRetail	GM $ (Retail)	The gross margin percent derived by calculating the cost of goods as follows: (sales dollars + markdown dollars) * (unit cost / original retail price)
gmPercentRetail	GM % (Retail)	The gross margin derived by calculating the cost of goods as follows: (sales dollars + markdown dollars) * (unit cost / original retail price)
markdownDollars	Markdown $	The forecasted markdown cost for the specified time period based on both permanent and temporary markdowns.
sellThroughPercentRemaining	Sell Thru %	The percentage of inventory sold through at the end of the specified time period.
inventoryUnits	EOH Units	The remaining units on hand at the end of the specified time period.

An example of a predefined row for the What If screen is Gross Margin Dollars Cost. This definition is found in the p4pgui-config.xml file.

```
<what-if-view-row
display-name="p4pgui.forecastWhatIfRow.gMDollarsCost.label"
description="p4pgui.forecastWhatIfRow.gMDollarsCost.description"
use-as="gMDollarsCost"
type="money"/>
Another, more complex, example is shown as follows. This example contains a format
attribute.
<what-if-view-row
display name="p4pgui.whatIfRow.priceLadder.label"
description="p4pgui.whatIfRow.priceLadder.description"
type="forecast-dropdown"
use-as="whatIfPrice"
format=p4pgui.whatIfRow.forecastPrice.format"/>
```

The following table describes these row attributes.

Table 10–29 Row Attributes for Recommended Forecast Screen

Tag	Description
display-name	Alias that points to the comparable label in the gridResources.properties file containing the label text that displays to the left of the row.
description	Alias that points to the comparable label description in the gridResources.properties file containing the context-sensitive label description for the row.
use-as	The calculation performed to produce the result displayed in the row. Acceptable values are: ladderID inventoryUnits inventoryDollars inventoryDollarsCost inventoryDollarsRetail sellThroughPercentRemaining sellThroughPercentTotal gMPercentCost gMDollarsCost gMPercentRetail gMDollarsRetail salesUnits salesDollars markdownDollars forecastPriceAverage forecastPriceExact forecastPriceAveragePercent forecastPriceExactPercent promoFlag

Table 10–29 (Cont.) Row Attributes for Recommended Forecast Screen

Tag	Description
type	Data type, for example, floating point or integer. Acceptable values are: ■ integer ■ money ■ percent ■ flag
format	Alias that points to the comparable formatting information in the gridResources.properties file. This formatting information determines the formatting of numeric data.

Configuring Other Features of the What If and Recommended Forecast Screens

In addition to configuring the rows and columns for the What If and Recommended Forecast screens, you also can configure the following screen features:

- Number of weeks to display
- Calculations such as:
  - price ladder
  - price
  - markdown cost
  - gross margin dollars
  - gross margin percent
  - sell-through percent
  - sales dollars and inventory dollars

Information about each of these settings is as follows:

Table 10–30 Recommended Forecast Row Metrics

Row Metric	Description
Price ladder	Contain the drop-down selection list (widgets) in which users specify the markdown selection. The configuration of price ladders is discussed in the section on advanced functional configuration. Note that you must ensure that the <grid-display> attribute for this metric is set at "allowed." Otherwise, the Recommended Forecast screen does not function.
Price	Prices recommended by the optimizer, that is, the suggested values of the register price. Note that the ticket price may be higher than the register price because of point-of-sale (POS) markdowns.
Markdown Cost	The value of the price change multiplied by the inventory on hand at the moment the markdown takes effect. Price change is defined as the difference between the last ticket price and the current register price.

Table 10–30 (Cont.) Recommended Forecast Row Metrics

Row Metric	Description
Gross Margin Dollars	Gross margin dollars calculated without regard to business rules or restrictions that define alteration costs and cash discounts. As a result, the What If screen may calculate markdowns that make the EOL gross margin larger than the recommended forecast.
Sell Through Percent	Percentage of inventory sold through a specified time period.
Sales Dollars	Dollar value of merchandise sold in a period.
Inventory Dollars	Dollar value of the on-hand inventory.

Configuring the User Administration and Edit User Screens

The User Administration screens, which consist of an initial User Administration screen that displays after login and an associated Edit User screen, enable the system administrator to:

- add or delete system users
- manage passwords
- determine access control by defining the following attributes for users:
  - Role definitions
  - Assignment to roles
  - Access to the application components
  - Access to the application (merchandise) items

The User Administration screens are shown in the following table.

Table 10–31 User Administration Screens

Screen	Access Method	Description
User Administration	Log in to the application.	Initial screen that displays after login using root password.
Edit User	On the User Administration screen, click **Edit**.	Has two works areas: • Modify Worksheets • Add Worksheets

Configuring the User Administration Screen

The user administration screen displays information about users, as shown in the following illustration.

The initial list of users displayed in the User Administration screen is specified in the following tags in the p4pgui-config.xml file. Note that the description and display-name attributes in these tags point to the gridResources.properties file, where the display text for these attributes is specified.

Note that typically you do not need to modify the following code for the User Administration screen.

```
<user-admin-list-column
id="999"
db-column-name="user_id"
use-as="UA_EDIT_DEL_BUTTONS"
```

```
db-table-name="p4p_user_info"
type="editdeletebutton"
sortable="false"/>
<user-admin-list-column
id="1000"
description="p4pgui.username.column.description"
display-name="p4pgui.username.column.label"
db-column-name="user_id"
db-table-name="p4p_user_info"
use-as="USERNAME"
sortable="true"/>
<user-admin-list-column
id="1001"
display-name="p4pgui.lastname.column.label"
description="p4pgui.lastname.column.description"
db-column-name="lastname"
db-table-name="p4p_user_info"
sortable="true"/>
<user-admin-list-column
id="1002"
display-name="p4pgui.firstname.column.label"
description="p4pgui.firstname.column.description"
db-column-name="firstname"
db-table-name="p4p_user_info"
sortable="true"/>
```

Configuring the Edit User Screen

The management hierarchies used by the retailer are reflected in the Edit User screen. Thus, if the retailer uses three hierarchies that correspond to the company, division, and department levels, you set up the Edit User screen to display three columns that correspond to these hierarchies.

The following table shows some commonly used predefined column definitions for the Edit User screen.

Table 10–32 Commonly Used Predefined Column Definitions for the Edit User Screen

Key	Display Name	Pop-Up Description
username	Username	The user's (case sensitive) login ID
lastname	Last Name	The user's last name
firstname	First Name	The user's first name
HIERARCHY1	Company	hierarchy1
HIERARCHY2	Division	hierarchy2
HIERARCHY3	Department	hierarchy3
lastMod	Last Modified	Date and time when changes were last made to the worksheet
modBy	Modified By	Name of user who last modified the worksheet
totalItems	Total Items	Total number of items in a worksheet
viewers	Viewers	Number of users who have View-Only access to a worksheet
submitters	Submitters	Number of users who have Submit access to this worksheet

Table 10–32 (Cont.) Commonly Used Predefined Column Definitions for the Edit User

Key	Display Name	Pop-Up Description
approvers	Approvers	Number of users who have Approve access to this worksheet
permission	Permission	Highest level of access that a particular user has for a worksheet

Configuring the Modify Worksheets Area You configure the Modify Worksheets area of the Edit User screen by modifying the <user-admin-user-summary-column> tags in the p4pgui-config.xml file.

The order in which the tabs are listed is the order in which the columns display from left to right in the Modify Worksheets area of the Edit User screen.

Configuring the Add Worksheets Area You configure the Add Worksheets area of the Edit User screen by modifying the <user-admin-nonuser-summary-column> tags in the p4pgui-config.xml file.

You make the configurations for this screen in the p4pgui-config.xml file.

Markdown Optimization Display-Only Screens

The application display-only screens, which display static text updates, cannot be configured at all. The following screens comprise the display-only type:

- Item information
- Promo details

Item Information Screen

The Item Info screen displays detailed information about a particular item. The retailer accesses this screen by clicking the underlined item label in the Description column in the Item Worksheet. The retailer can select links on the Item Info screen to view pages with even more detailed information.

To configure this screen, you insert the data fields using the following files:

- item-details-layout.xml
- p4p-column-list.xml

Promo Details

Promo details is a popup window associated with the What If screen. This window provides additional details about promotions that are not available on the What If screen. To display this window, click Promo in the Promo Flag row of the What If screen.

Typically, you accept the default configuration found in the p4p-promo-details-grid.xml file.

This file, which uses the same syntax as the XML column configuration files, is shown as follows.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
Sample XML file generated by XML Spy v4.4 U (http://www.xmlspy.com)
-->
```

```
<grid xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Grid.xsd" key="p4p-promo-details-grid">
<grid-properties name="p4pgui.grid.label.EditItemWorksheet" defaultrowlevel="1" />

<column-group-spec>
<column>
<key>INT_P_ITEM_ID</key>
<parent-key />
<column-properties display-type="integer" />
</column>
<column>
<key>P_PROMO_DESC</key>
<parent-key />
<column-properties />
</column>
<column>
<key>P_OFFER_LADDER_RUNG</key>
<parent-key />
<column-properties />
</column>
<column>
<key>P_OFFER_LADDER_PCT</key>
<parent-key />
<column-properties />
</column>
<column>
<key>P_START_DT</key>
<parent-key />
<column-properties />
</column>
<column>
<key>P_END_DT</key>
<parent-key />
<column-properties />
</column>
</column-group-spec>
<row-group>
<key>INT_P_ITEM_ID</key>
<rowgroup-properties expandable="true" isexpanded="true">
<groupby>INT_P_ITEM_ID</groupby>
</rowgroup-properties>
<summary-info />
</row-group>
<summary-info />
</grid>
```

Price Ladders

This section provides information about configuring price ladders.

Data Sources and Pre-configured Properties for Price Ladders

The following database views serve as data sources for the price ladders.

The properties for the price ladder rungs are pre-configured and have been loaded into the database tables. This means that you need to perform only minimal configuration for the price ladders.

The rung properties that are already loaded into the database are:

- dollar amount or percentage
- price ladder name
- price ladder type

Configuring Price Ladder Properties

Your configuration activities for price ladders consist of verifying that the following specifications are made in the appropriate files. Note that these specifications come pre-configured with the baseline application code and should be present in the appropriate files unless a team member from your installation has modified the files in question.

The specifications you need to verify are shown in the following table:

Table 10–33 Price Ladder Specifications

File Name	Specifications to Verify	Description
p4p-column-list.xml	The following column definitions: INT_PROPOSED_PRICE INT_LADDER_ID INT_TICKET_PRICE INT_LADDER_INITIAL_VAL	 Markdown price display column Markdown type display column Ticket price not including promotions Non-displayed column
p4p-[grid-name]-grid.xml	The following column dependency tags: <ColumnDependency>INT_LADDER_INITIAL_VAL</ColumnDependency> <ColumnDependency>INT_TICKET_PRICE</ColumnDependency>	These tags ensure that the needed calculations are performed on the price ladder metrics. You must ensure that these tags are present in the XML grid files for that application screen.
NA	The column keys that map to the required column keys in the XML grid files. These are: INT_PROPOSED_PRICE (markdown price display column) INT_LADDER_ID (markdown type display column)	You must ensure that these columns are correctly configured for the hierarchy level at which they are to display on the screen (parent or leaf level). The code for both parent and leaf-level column definitions is shown as follows.

Following are column definitions in a p4p-[grid-name]-grid.xml file for price ladders that display at the parent level.

```
<ColumnDependency>INT_LADDER_INITIAL_VAL</ColumnDependency>
<grid-properties
<ColumnDependency>INT_TICKET_PRICE_WT_AVG</ColumnDependency>
</grid-properties>
<column-group-spec>
<column>
<key>INT_PROPOSED_PRICE</key>
<parent-key>INT_PROPOSED_PRICE</parent-key>
<column-properties
type="double"
editable-in-readmode="false">
<function key="P4P_PRICELADDER/">
```

```
</column-properties>
</column>
<column>
<key>INT_LADDER_INITIAL_VALUE</key>
<parent-key>INT_PROPOSED_PRICE</parent-key>
<column-properties
visibility="not-visible"
orderable="false"
hideable="false">
<function key="P4P_PRICELADDER/">
</column-properties>
</column>
Following are column definitions in a p4p-[grid-name]-grid.xml file for price
ladders that display at the leaf level.
<grid-properties
<ColumnDependency>INT_LADDER_INITIAL_VAL</ColumnDependency>
<ColumnDependency>INT_TICKET_PRICE_WT_AVG</ColumnDependency>
<ColumnDependency>INT_COLLECTION_NAME</ColumnDependency>
</grid-properties>
<column-group-spec>
<column>
<key>INT_PROPOSED_PRICE</key>
<parent-key>INT_PROPOSED_PRICE</parent-key>
<column-properties
type="double"
editable-in-readmode="false">
<function key="P4P_PRICELADDER"/>
</column-properties>
</column>
<column>
<key>INT_LADDER_ID</key>
<column-properties
displaytype="dropdown"
editable-in-readmode="false"
read-only type="static-text"
<function key="P4P_LADDERPICKER"/>
</column-properties>
</column>
<column>
<key>INT_LADDER_INITIAL_VAL</key>
<parent-key>INT_PROPOSED_PRICE</parent-key>
<column-properties
visibility="not-visible"
orderable="false"
hideable="false>
<function key="P4P_AVG">
<args>INT_INVENTORY</args>
</column-properties>
</column>
```

11

Configuring the Application (GUI)

This chapter describes the configuration of the MDO application user interface.

It contains the following sections:

Introduction

This chapter describes how to set up the Markdown Optimization application, which consists of the user interface and those software and hardware components that support it.

Note: The Grid Designer is a Web-based rich application that enables you to modify the following main components of the grid configuration in a graphical user interface:

- Column definitions
- Grid resources
- Grid configuration files
- Report configuration files

For more information on the Grid Designer, refer to the *Oracle Retail Markdown Optimization Grid Designer User Guide.*

Overview of the Markdown Optimization Application Configuration Process

This section provides an overview of the application configuration process.

Completing Pre-Configuration Requirements

Before you begin the application configuration, you must ensure that these basic pre-configuration activities have been completed:

- The retailer's business requirements have been gathered and captured on an Excel spreadsheet.
- The following key tables of the application database have been configured and populated with at least a minimal set of data – either sample data or a subset of the retailer's data:
  - ITEM_DATA
  - PRICE_LADDERS_TBL
  - P4P_SUBMITTAL_WORKSHEETS

  This data load is needed because Markdown Optimization is a data-driven application that cannot function without data.

High-Level View of Configuration Tasks

Here is a high-level view of all possible tasks that you might perform to configure each user interface screen. The scope of configuration tasks that you do for a given screen depends on its configuration type — total, limited, or display-only. You perform all of these tasks for total configuration type screens and only a subset of them for the other configuration types.

The following tasks are listed in the order in which you perform them; each of them is described in detail later in this chapter.

1. Set up your workstation and user permissions.
2. Set up the screen hierarchy levels.
3. Configure the limited configuration type screens.
4. Configure the display-only type screens.
5. Set up the user administration features.
6. Perform advanced functional configuration for Price Ladders, Budget, and Markdown Accounting.

Setting Up the Workstation and User Permissions

Once you have ensured that the retailer business requirements have been captured and the application database tables have been loaded with the needed data, the next steps are to:

1. Load the required software onto your local workstation.
2. Obtain user administrative permissions for yourself.

Setting Up the Hierarchy Levels

After setting up your local workstation and obtaining the needed user access, your next step is to set up the hierarchy levels. You must set up the hierarchy levels before you can begin configuring the total configuration type screen.

The three major steps for setting up the hierarchy levels are:

1. Determine the retailer hierarchy levels.
2. Set up the hierarchy information in the appropriate XML configuration files.

   After editing the files, it is helpful to search them for the string HIERARCHY to ensure that you have properly set up the hierarchy levels.

3. Reboot WebLogic and test the setup.

Determining the Hierarchy Levels

There are two options for determining the retailer hierarchy levels, depending on whether the retailer data has been already loaded into P4P_SUBMITTAL_WORKSHEETS.

To determine the retailer hierarchy levels:

- If the retailer data has not been loaded:

  Examine the application metrics spreadsheet that contains the retailer's business requirements.

- If the retailer data has already been loaded:

  Use Rapid SQL or another similar tool to view a snapshot of the database. Here is a view of the data from P4P_SUBMITTAL_WORKSHEETS as viewed through RapidSQL.

  In this view, each column represents a column on the grid display. Note that the HIERARCHY1 and HIERARCHY2 columns contain data representing department or division numbers as defined by the retailer. The presence of data in these columns means that the data for these department or divisions numbers are displayed on the grid. The absence of data in the HIERARCHY3 and HIERARCHY4 columns means that these hierarchies do not display on the grid.

Configuring the Hierarchy Levels

After you determine the hierarchy levels, configure the appropriate hierarchy elements in the appropriate XML configuration files. The files and elements to configure are shown in the following table.

Table 11–1 XML Configuration Files for Hierarchy Updates

File	Hierarchy Elements to Configure
p4p-custom-columns.xml	INT_WKSHT_HIERARCHY INT_STYLE_DESC
p4pgui-config.xml	hierarchy-html-form-name= worksheet-params findKey=
p4pgui-wksht-summary-grid.xml	<column> <key> row-group override-column-group

An example for each of these three file types is shown as follows.

Example of p4p-custom-columns.xml File

This example shows how the INT_WKSHT_HIERARCHY and INT_STYLE_DESC elements are configured to set up the hierarchy levels.

```
W<?xml version="1.0" encoding="UTF-8" ?>
- <!--
 edited with XML Spy v4.3 U (http://www.xmlspy.com)
--> - <columnlist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="Grid.xsd" key="customColumns">
- <!--
```

```
 START : Columns overridden from p4p-columns-list.xml
--> - <column-def>
<key>INT_WKSHT_HIERARCHY</key>
<column-def-properties label="p4pgui.HIERARCHY4.column.label"
description="p4pgui.HIERARCHY4.column.description" db-table-name="p4p_submittal_
worksheets"
db-column-name="HIERARCHY4" groupId="GROUP_HEADER" filterable="true"
sortable="true"
orderable="true" operatortype="equals" />
</column-def> - <column-def>
<key>INT_STYLE_DESC</key>
- <column-def-properties composeable="false" label="p4pgui.HIERARCHY8_
NAME.column.label"
 type="string" db-column-name="HIERARCHY8_NAME"
description="p4pgui.HIERARCHY8_NAME.column.description" db-table-name="P4P_
DISPLAY_ITEMS"
filterable="true" sortable="true" orderable="true" hideable="false"
expandable="false" filtertype="text"
operatortype="equals" groupId="GROUP_HEADER">
<function key="P4P_SAME_OR_NULL" />
</column-def-properties>
</column-def>
- <!--
 END : Columns overridden from p4p-columns-list.xml
-->
- <!--
```

Example of p4pgui-config.xml File

This example shows how the hierarchy html-form-name and worksheet-params findKey elements are configured to set up the hierarchy levels.

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE xml PUBLIC "-//DTD p4pgui config//EN" "p4pgui-config.dtd">
<xml>
.
.
<hierarchy html-form-name="hierarchy1" id="1" key="HIERARCHY1"/>
<hierarchy html-form-name="hierarchy2" id="3" key="HIERARCHY2"/>
<hierarchy html-form-name="hierarchy3" id="5" key="HIERARCHY3"/>
<hierarchy html-form-name="hierarchy4" id="7" key="HIERARCHY4"/>
<hierarchy html-form-name="hierarchy5" id="9" key="HIERARCHY5"/>
<hierarchy html-form-name="hierarchy6" id="11" key="HIERARCHY6"/>
<hierarchy html-form-name="hierarchy7" id="13" key="HIERARCHY7"/>
<hierarchy html-form-name="hierarchy8" id="15" key="HIERARCHY8"/>
.
.
<worksheet-params findKey="HIERARCHY8" allow-sendback-date="true"/>
  <merchandise-maint-params endingInv-input-type="tgtSellThruPerc">
    <items>
      <outdate-constraints range="365">
        <excluded-days/>
      </outdate-constraints>
    </items>
    <collections>
      <outdate-constraints range="365">
        <excluded-days/>
      </outdate-constraints>
    </collections>
  </merchandise-maint-params>
  <page name="worksheet">
```

```
    <view gridname="collectionsGrid"/>
    <view gridname="itemsGridGroupStyle"/>
    <view gridname="itemsGridFlat"/>
  </page>
  <page name="merchandise">
    <view gridname="maintGridCollections"/>
    <view gridname="maintGridFlat"/>
  </page>
  <export-filenames mkdn-to-client="pma_mkdn" mkdn-to-pl="pma_mkdn_pl"
outdates-to-pl="pma_outdate_pl"/>
  <hints on="true">
    <hint name="RULE">RULE</hint>
    <hint name="ORDERED">ORDERED</hint>
  </hints>
</xml>
```

Example of p4p-wksht-summary.xml File

This example shows how the column, row-group, and override-column-group elements are configured to set up the hierarchy levels.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
 edited with XML Spy v4.3 U (http://www.xmlspy.com)
--> <column>
<key>ROWSELECT</key>
<parent-key>SELECT_ROW</parent-key>
<column-properties />
</column>
.
.
.
- <!--
//
--> - <row-group>
<key>CHAIN</key> - <rowgroup-properties expandable="true" isexpanded="true"
name="ALL">
<groupby>EMPTY</groupby>
</rowgroup-properties> -
<override-column-group> -
 <column>
<key>INT_WKSHT_HIERARCHY</key>
<column-properties display-type="blank" />
</column> - <column>
</override-column-group> - <row-group>
<key>WKSHT_HIERARCHY4</key>
<rowgroup-properties expandable="true" isexpanded="true" />
<override-column-group />
</row-group>
</row-group>
</grid>
```

Configuring Total Configuration Type Screens

Because total configuration type screens contain the largest number of elements to configure, they are the most complex to work with. Therefore, it is best to begin the configuration process by configuring screens of this type first. The majority of your application configuration effort consists of configuring these screens.

The total configuration type screens are as follows:

- Worksheet summaries
- Worksheets
- Edit item worksheet
- Maintaining merchandise screens
- Pricing Groups
- Edit pricing groups
- Reports

The Worksheet Summaries screen serves as a good example of how to configure the total configuration type screen because of the following:

- This screen is the first screen that the user sees upon startup. Therefore, for optimal testing, this screen should be working first.
- Most of the other screen types are built upon it.
- The Worksheet Summaries screen has the most complex configuration of all the application screens.

Configuring a Sample Screen

In this section, the configuration of the Worksheet Summaries is described by reviewing the configuration procedures for a screen.

Here are the high-level steps for configuring a screen. These steps are described in detail in the following pages:

1. Identify the application screen metrics.
2. Identify the data source.
3. Create the columns.
4. Configure the grid features.
5. Test the display.

Identifying the Application Screen Metrics

Your first step in the configuration process is to determine the screen metrics.

The columns in this spreadsheet is described in the following table.

Table 11–2 Application Metrics Spreadsheet Columns

Column	Header	Description
A	Use	Whether the column is used in the grid – Yes or No
B	Column Key	Unique identifier for the column
C	Display Name	Label to appear at the top of the column on the screen display
D	Pop-Up Description	Context-sensitive description that appears when the user selects the column label

Identifying the Data Source

To configure the Worksheet Summaries screen, you must determine the data source for each column in the grid. The data source types are:

- Direct metric – direct reference to a database column
- Derived metrics:
  - Simple derivation – derived from an operation on a database metric
  - Complex derivation – derived from a user-defined column that is derived from a database column

To create columns for the Worksheet Summaries screen, you must identify and specify data sources in both of the XML column files — p4p-column-list.xml and p4p-custom-columns.xml. You must use both column files because you need to configure two columns for each column that appears on the Worksheet Summaries screen.

In the column files, you specify the data source within the <column-def-properties> element. The following table shows:

- The data sources for the two columns that you configure to create the # Rec MD column, which is the sample column described in this example.
- The configuration files in which these data sources are defined

Table 11–3 Data Sources and Configuration Files for Worksheet Summaries Columns

Database Table Name	Database Column Name	XML Configuration File
P4P_DISPLAY_ITEMS	RECOMMENDED_ITEM_FLAG	p4p-column-list
p4p_submittal_ worksheets	NUM_RECOMMENDED_MARKDOWNS	p4p-custom-columns

Creating Columns

Columns are the basic building blocks of the application screens. Each column, as defined in the application metrics spreadsheet represents one metric (type of data) for each row. When you configure the screens, you must define both displayed and hidden columns.

You do not define the standard, out-of-the-box application columns, which are already configured.

Configuring Multiple Columns in the Spreadsheet

The sample screen configuration described in this section shows the configuration of only one column — # Rec MD (Number of Recommended Markdowns). Once you understand how this column is created, you can use the same process to create the other needed columns for the screen.

Typically, when configuring a Markdown Optimization screen, you define several related columns at a time and then test them.

Creating the #Rec MD Column

Creating the # Rec MD column in the sample Worksheet Summaries screen requires defining elements in five different Markdown Optimization configuration files. Each of these files has elements that map to the comparable elements in the other files used for the column configuration.

The Worksheet Summaries screen aggregates and displays data from other screens, specifically, from the Items Worksheet screens. That is, this screen shows metrics (item-level calculations) that do not exist at the Item Worksheet level. Because of this

aggregation of data from other screens, you must configure two columns for each column that appears on the Worksheet Summaries screen.

The following table shows the files used in configuring the # Rec MD column and the metrics configured in each file. These files are listed in the suggested order of configuration:

Table 11–4 Configuration Files and Metrics to Define # Rec MD Column

File	Metrics
1. p4pgui-config.xml	Worksheet-level variable
	Column key reference
	Aggregation type
2. p4p-column-list.xml	Column key
	Data source definition
	Column display features
3. p4p-custom-columns.xml	Custom column to display variable
	Data source definition
	Alias for column label text
	Alias for context-sensitive column label description
	Column display features
4. p4p-wksht-summary-grid.xml	Column key This section describes only the column-specific configurations for this file.
5. gridResources.properties	Display text for column label
	Display text for context-sensitive column label description.

For each configuration file, the following sections describe the:

- Configurations to specify in this file
- Procedures for specifying each of these configurations

Configuring p4pgui-config.xml

This file is the best place to begin the column configuration because it is where you set up the hierarchies.

To configure this file, define the following within the <metrics> element:

- Create a worksheet-level variable
- Populate this variable by defining a reference to the column key
- Specify a formula for this variable

To create the worksheet-level variable, create a metric that defines the number of recommended markdowns, as follows:

```
<metrics>
<metric-items
name = "num_recommended markdowns"
</metrics>
```

This metric points to the corresponding column metric in the p4p-custom-columns.xml file, which in turn points to the database.

As you continue defining the screen grid, create a new worksheet-level variable (column metric) for each column.

The <items-metric> element in p4pgui-config.xml is used to specify the footnote summary metrics. The valid type attributes for the <items-metric> element are date, numeric, percent, money, and status. Here is a configuration example that includes a custom format. If the type is numeric and no format is specified, then the format defaults to the suite-wide integer format. In order to get a decimal format, a specific format must be supplied.

```
<items-metric id="oh" display-name="p4pgui.metrics.oh.label" type="numeric"
format="p4pgui.metrics.format.numeric" row="3" column="1"/>
```

To populate this worksheet-level variable, create a reference to the corresponding column key, as follows:

```
<metrics>
<metric-items
column-ref="INT_RECOMMENDED_ITEM_FLAG"
</metrics>
```

In this example, the corresponding column key is contained in the p4p-column-list.xml file. (In other cases, the key might be found in the p4p-custom-columns.xml file.) By creating this reference you are creating a link between the worksheet and the item.

To define the aggregation type, enter an acceptable value for this element.

Note that when using the summary metrics tag, the aggregation type is a required field.

In this example, the aggregation type is defined as follows:

```
<metrics>
<metric-items
aggregationType = "SUM"
</metrics>
```

The aggregation type in this example defines the value of "num_recommended markdowns" as the sum of INT_RECOMMENDED_ITEM_FLAG.

Here is the code in the p4pgui-config.xml file to use to review the configured file after all the preceding elements have been configured.

```
<metrics>
<metric-items
name = "num_recommended markdowns"
column-ref="INT_RECOMMENDED_ITEM_FLAG"
aggregationType = "SUM"
</metrics>
```

Configuring p4p-column-list.xml

This file contains the configurations for the following Worksheet Summaries screen column metrics:

- Column key
- Data source definition
- Column display features

To create the column key (unique identifier for the column), define the key in the <column-def> element, as follows:

```
<key>INT_RECOMMENDED-ITEM-FLAG</key>
```

This is an item-level piece of information.

This key is referenced by the column reference in the p4pgui-config.xml file as follows:

```
column-ref="INT_RECOMMENDED_ITEM_FLAG"
```

To create the data source definition (that is, to assign a value to the int_ recommended_ item_flag variable), specify the database table name and column name within the <column-def-properties> element, as follows:

```
<column-def>
<column-def-properties
db-table-name="P4P_DISPLAY_ITEMS"
db-column-name="RECOMMENDED_ITEM_FLAG"/>
</column-def>
```

The result of creating this data source definition is that the column reference INT_ RECOMMENDED_ITEM_FLAG (defined in the p4pgui-config.xml file) takes the corresponding value in the database.

To define the column display features, assign values to the column attributes, as follows:

```
<column-def>
<column-def-properties
type="integer"
display-type="integer"
orderable="false/>
</column-def-properties>
</column-def>
```

For information on acceptable values for the <column-def-properties> element, see Table.

Note that if the display type attribute of a column is date, the date displays in the same format as the operating system's date.

Here is the configuration of the p4p-column-list.xml file to use to review the configured file after all the preceding elements have been defined.

```
<column-def>
<key>INT_RECOMMENDED_ITEM_FLAG</key>
<column-def-properties
db-table-name="P4P_DISPLAY_ITEMS"
db-column-name="RECOMMENDED_ITEM_FLAG"
type="integer"
display-type="integer"
orderable="false/>
</column-def>
```

Note that the INT_TICKET_PRICE and the INT_TICKET_PRICE_WT_AVG metrics specified in p4p-column-list.xml refer to PERM_TICKET_PRICE (the second derivation), but they have different aggregations. INT_TICKET_PRICE is an internal metric that is used to truncate price ladder drop-down values, so the maximum value in the price ladder drop-down menu will not exceed the minimum ticket price in the grouping. This could happen, for example, when the price ladder is at the pricing group level and there is a group of items below that level. This is done to disallow

markups. You should not override the aggregation type for INT_TICKET_PRICE. It must always be of type MIN. In addition, it should be hidden, because the ticket price that is usually visible to the retailer has an average type of aggregation weighted by the inventory (see INT_TICKET_PRICE_WT_AVG) not MIN. To display a ticket price, INT_TICKET_PRICE_WT_AVG should be used and it should be made visible.

Configuring p4p-custom-columns.xml

This file contains the configurations for the following Worksheet Summaries screen column metrics:

- Custom column to display the variable
- Data source definition
- Alias for the column label text
- Alias for the context-sensitive label description
- Column display features

To define a custom column to display this variable that you have created for the worksheet, create a column key as follows:

```
<key>NUM_REC</key>
```

This key name must be unique because it serves as the unique identifier for the # Rec MD column.

To create the data source definition (that is, to assign a value to the num_recommended_markdowns variable), specify the database table name and column name within the <column-def-properties> element, as follows:

```
<column-def>
<column-def-properties
db-table-name="p4p_submittal_worksheets"
db-column-name="NUM_RECOMMENDED_MARKDOWNS"/>
</column-def-properties >
</column-def>
```

This data source definition acts as an intermediary between the database and the corresponding definition in the p4pgui-config.xml file.

That is, this definition points directly to the database, from which it takes the corresponding value for that variable. From the other end, the corresponding element in the p4pgui-config.xml file points to this definition. Therefore, the metric-items name = "num_recommended markdowns" element in the p4pgui-config.xml file gets populated with data by referring to this column reference.

To define an alias (pointer) to the column label text, create the following label definition:

```
label = "p4pgui.numRec.column.label"
```

This label definition, which points to the gridResources.properties file, contains the actual label text to display at the top of the column.

To define an alias (pointer) to the context-sensitive label description for the column, create the following label definition:

```
description = "p4pgui.numRec.column.description"
```

This label definition, which points to the gridResources.properties file, contains the text that displays when the user hovers the mouse over the label text at the top of the column.

To define the column display features, assign values to the column attributes, as follows:

```
<column-def>
<column-def-properties
display-type="integer"
orderable="true"
sortable="true"
hideable="true
</column-def-properties>
</column-def>
```

Here is the configuration of the p4p-custom-columns.xml file to use to review the configured file after all the preceding elements have been defined.

```
<column-def>
<key>NUM_REC</key>
<column-def-properties
db-table-name="p4p_submittal_worksheets"
db-column-name="NUM_RECOMMENDED_MARKDOWNS"/>
label = "p4pgui.numRec.column.label"
description = "p4pgui.numRec.column.description"
display-type="integer"
orderable="true"
sortable="true"
hideable="true
</column-def-properties>
</column-def>
```

Configuring p4p-wksht-summary-grid.xml

To get the column properties, this file contains a reference to the NUM_REC column key in the p4p-custom-columns.xml file.

In this example, this reference is defined as follows:

```
<column-properties>
<key>NUM_REC</key>
</column-properties>
```

When the p4p-wksht-summary-grid.xml file contains column property definitions, these definitions override the ones defined in either of the column files.

Configuring gridResources.properties

This file contains configurations for:

- The label that appears at the top of the # Rec MD column
- The description that displays when the user hovers the mouse over the column label display

The text label and description are defined in this file as follows:

p4pgui.numRec.column.label = # Rec MD

p4pgui.numRec.column.description = Number of recommended markdowns

The Mapping Between the Column Configuration Files

Each of the five column-related configuration files contains elements that map to the comparable elements in the other files used for the column configuration. The mapping of these elements from one file to the other is shown in the following illustration.

Configuring the Grid Features

After the columns are configured, the next step is to configure the grid.

Each Markdown Optimization grid screen enables you to group columns of information. To prevent blank columns being grouped together, it is recommended that you enable grouping only on required fields.

The p4p-wksht-summary-grid.xml file for the sample Worksheet Summaries screen used in this section specifies:

- The column group specification, which determines:
  - The columns to display on the grid
  - The display order of these columns

  The column specifications in this file do not include column configurations such as data source, display properties, and function keys. These properties are specified in the <column-def-properties> element in the application XML column files.
- The row group specification, which determines:
  - The nesting of the rows
  - The different formats in parent and child rows
  - If applicable, specifications that override column group settings
- Identifying, functional, and display properties for the row and column groups

Configuring Maintaining Merchandise Grids

In the Maintaining Merchandise grids, P4P_DISPLAY_ITEMS has been replaced by P4P_MAINTAIN_ITEMS. Maintaining Merchandise grids do not use the P4P_DISPLAY_ITEMS view as db-table-name. Instead, they use the P4P_MAINTAIN_ITEMS view.

The following metrics are only available on the Maintaining Merchandise views. These metrics must have the following property set for them to filter on (which is in the Maintaining Merchandise grids):

```
<custom-property name= "useMaintainView" value= "true" custom-type=
"application"/>
```

Metric Name	Metric Key
New Out Date	INT_MOD_OUTDATE
New Salvage Value%	INT_MOD_SALVAGE_VAL_PERC
New Sell Through%	INT_MOD_INV_TARGET_ST_PERC
New Ending Inventory	INT_MOD_INV_TARGET_END_UNITS
New Start Date	modifiedStartDate

Specifying the Columns That Comprise the Grid

The p4p-wksht-summary-grid.xml file for the sample Worksheet Summaries screen used in this section specifies the columns that comprise the grid. The column group specification in this grid file contains the twelve columns that display on the screen. It also contains one column that is not visible on the screen.

The columns and their keys are shown in the following table. Except for the standard columns (select row, expand-collapse, worksheet ID, and worksheet status),

specifications for all these columns can be found in the business requirements spreadsheet.

Table 11–5 Columns Specified in Sample p4p-wksht-summary-grid.xml File

Column Name	XML Element or Attribute	Additional Information
Select row	ROWSELECT	Check box for selecting the row.
Expand-Collapse	EXPCOL	Additional column to allow expansion and collapsing. Required if any of the columns have the value "true" in the expandable attribute.
Worksheet ID	INT_WORKSHEET_ID	For internal use.
Chain	HIERARCHY1	Not displayed in the sample Worksheet Summaries screen.
Division	HIERARCHY2	NA
Department	HIERARCHY3	NA
Status	WKSHT_STATUS	NA
Number of recommended markdowns		Sample column used as configuration example for this worksheet.
Recommended markdown dollars	REC_MARKDOWN_DOLLARS	NA
Recommended markdown dollar cost	REC_MARKDOWN_DOLLAR_COST	NA
Taken markdown dollars	TAKEN_MARKDOWN_DOLLARS	NA
Taken markdown dollar cost	TAKEN_MARKDOWN_DOLLAR_COST	NA
Number of added markdowns	NUM_ADDED_MARKDOWNS	NA
Number of taken markdowns	NUM_TAKEN_MARKDOWNS	NA

Here are the column specifications in the p4p-wksht-summary-grid.xml file for the sample Worksheet Summaries spreadsheet. These specifications indicate:

- Which columns (both visible and non-visible) should be included in the Worksheet Summaries screen.

  Note that every column key in this file must point to the identical key in one of the column files. If the column key is not correct, the column will not display on the screen.

- The order in which these columns display on the screen.

  The first column specified in this file indicates the left-most column that displays on the screen, and so on.

```
<?xml version="1.0" encoding="UTF-8" ?>
<grid xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Grid.xsd" key="summary">
```

```
<grid-properties name="grid" defaultrowlevel="1" firstrowheadercolumn="2"
columnsFrozen="5" filterable="false" />
<column-group-spec>
<column>
<key>ROWSELECT</key>
<parent-key>SELECT_ROW</parent-key>
<column-properties/>
</column>
<column>
<key>EXPCOL</key>
<parent-key>EXPCOL</parent-key>
</column>
<column-properties/>
<column>
<key>INT_WORKSHEET_ID</key>
<column-properties/>
</column>
<column>
<key>HIERARCHY1</key>
<parent-key/>
<column-properties
visibility=”not-visible”
hideable=”false”
orderable=”true”
sortable=”true”/>
</column>
<column>
<key>HIERARCHY2</key>
<parent-key/>
<column-properties
type=”double”
orderable=”true”
sortable=”true”
display-type=”integer/>
</column>
<column>
<key>HIERARCHY3</key>
<parent-key/>
<column-properties
type=”double”
sortable=”true”
orderable=”true”
display-type=”integer/>
</column>
<column>
<key>WKSHT_STATUS</key>
<parent-key>WKSHT_STATUS</parent-key>
<column-properties
orderable=”true”
sortable=”true”
hideable=”true”/>
</column>
<key>NUM_REC</key>
<column>
<parent-key/>
<column-properties/>
</column>
<column>
<key>REC_MARKDOWN_DOLLARS</key>
<parent-key/>
```

```
<column-properties/>
</column>
<column>
<key>REC_MARKDOWN_DOLLAR_COST</key>
<parent-key/>
<column-properties/>
</column>
<column>
<key>TAKEN_MARKDOWN_DOLLARS</key>
<parent-key/>
<column-properties/>
</column>
<column>
<key>TAKEN_MARKDOWN_DOLLAR_COST</key>
<parent-key/>
<column-properties/>
</column>
<column>
<key>NUM_ADDED_MARKDOWNS</key>
<parent-key/>
<column-properties/>
</column>
<column>
<key>NUM_TAKEN_MARKDOWNS</key>
<parent-key>PARENT_KEY</parent-key>
<column-properties/>
</column>
</column-group-spec>
```

Specifying Row Hierarchies in the Grid

Typically, you configure worksheets so that some rows display summarized data from other rows. For example, a worksheet might contain a set of adjoining rows that display the data for several different colors of the same item, with each row displaying the data for a different color. To display the aggregated data from all the different item colors, you create a summary row.

The row hierarchies that you create are shown in the following table.

Table 11–6 Row Hierarchies in the Grid

Definition	Display Data
Parent-level row	Summary data, that is, a defined aggregation of database records.
Child-level row	Item-level data, that is an individual database record.

Markdown Optimization enables you to define multiple levels of child rows. Child rows are nested within the next highest level of row, which may be either the parent row or a higher-level child row. Each row represents either a record in the database or a defined aggregation of records.

Aggregated Data View

Data views in Markdown Optimization can be configured to display items in various ways. Items can be aggregated so that merchandise is displayed at the style level (or another level, depending on the implementation) instead of the item level.

To configure an aggregated view:

- Modify p4pgui-config.xml as follows:

In <page name = “worksheet”>, enter <view gridname=”aggregated-items-grid”/>

where gridname is the key found in p4p-aggregated-grid.xml.

- The default value for max-worklist-query in p4pgui-config.xml, which determines the maximum number of rows that are displayed in a grid, is set to 500 for maximum performance.
- Set the max-visible-columns in the <grid-properties> section of any grid file to 30 to maximize performance.

Specifying Other Grid Attributes

The XML grid file is where you define properties for the grid as a whole. These properties include:

- Rows and columns frozen
- Default row level
- Filtering properties
- First-row header column
- Grid name

The following excerpt from the wksht-summaries-grid.xml file for the sample worksheet shows the grid properties for the sample Worksheet Summaries screen.

```
<grid xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Grid.xsd"
key=”summary”>
<grid-properties
name=”grid
defaultrowlevel=”1”
firstrowheadercolumn=”2”
columnsFrozen=”2”
filterable=false/>
```

Configuring Limited Configuration Type Screens

When you configure limited configuration screens, you choose from a predefined selection of rows and columns. Because of these predefined elements, the configuration process for limited configuration type screens is much simpler than for the total configuration type screens. The predefined rows and columns are specified in the p4pgui-config.xml file, which is located in the grids folder.

The following screens are limited configuration screens:

- What If
- Recommended Forecast
- User profile
- Administration

Configuring the What If Screen

Configuring the What If screen involves the following high-level steps:

1. Identify the screen metrics.
2. Verify that the database tables are populated with data.

3. Make local copies of the configuration files on your workstation.
4. Edit the configuration files.
5. Save the files under source control.
6. Test the configuration.

Identifying the Screen Metrics

To identify the screen metrics for the What If screen, locate the following metrics in the business requirements spreadsheet:

- Required rows
- Required columns
- Other metrics such as:
  - Forecast parameters such as number of forecast weeks.
  - Display parameters such as price ladders, markdown cost, and gross margin dollars.

Verifying Data in Database Tables

Make sure that the following database tables are populated with data:

- FORECAST_ACTIVITIES
- ITEM_DATA

Copying, Editing, and Saving the Configuration Files

To copy, edit, and save the configuration files:

1. Ensure that you have made a copy of the p4pgui-conf.xml file on your local workstation.
2. In this copy of p4pgui-config.xml, identify the row and column definitions, which are as follows:

   <what-if-view-column>

   [properties]

   <what-if-view-row>

   [properties]
3. Do either of the following to the row and column definitions that you do not plan to use for the implementation:
   - Delete them.

     or
   - Insert comment delimiters around them so that the application does not read them, for example:

```
<!--forecast-view-row
display-name="p4pgui.forecastWhatIfRow.salesDollars.label"
description="p4pgui.forecastWhatIfRow.salesDollars.description"
use-as="salesDollars"type="money"
format="p4pgui.forecastWhatIfRow.gMDollarsRetail.format/>-->
```

After you complete this step, the only functioning row and columns definitions left in the p4pgui-conf.xml file should be those to be used in the implementation.

4. Save the file and, when appropriate, check it into source control.

Testing the configuration

Test the screens for proper display of the selected metrics.

Configuring the Recommended Forecast screen

The steps for configuring the Recommended Forecast screen are as follows:

1. Identify the screen metrics.
2. Ensure that the ITEM_DATA database table is populated with data.
3. Identify the required rows, columns, and other metrics (such as number of weeks to display) specified in the business requirements spreadsheet for the Recommended Forecast screen.
    1. Use the following syntax for defining rows and columns.

        <what-if-view-column>

        [properties]

        <forecast-view-row>

        [properties]
4. In the p4pgui-config.xml file, identify the available columns and rows.

    An example of a What If column definition is as follows:

    ```
    <what-if-view-column
    display name="p4pgui.whatIfCol.TTOOS.label"
    description="p4pgui.whatIfCol.TTOOS.description"
    use-as="TTOOS"/>
    ```

    If the business requirements spreadsheet specifies rows or columns that are not listed in the p4pgui-config.xml file, see the Contract Solutions Manager or Business Consultant for the retailer installation.
5. Make a copy of the p4pgui-conf.xml file on your local workstation.
6. In this copy of p4pgui-config.xml, do either of the following to the row and column definitions that you do not want to use for the implementation:
    - Delete them.
    - Insert comment delimiters around them, for example:

        ```
        <!--forecast-view-row
        display-name="p4pgui.forecastWhatIfRow.salesDollars.label"
        description="p4pgui.forecastWhatIfRow.salesDollars.description"
        use-as="salesDollars"type="money"
        format="p4pgui.forecastWhatIfRow.gMDollarsRetail.format/>-->
        ```

        In other words, after completing this step, only those row and column definitions to be used for the implementation should be left in the p4pgui-config.xml file.
7. Save the file and, when appropriate, check it into source control.

8. Test the screens for proper display of the selected metrics.

12

Configuration Properties Files

This chapter lists the MDO configuration settings and contains the following sections:

Introduction

The configuration properties files enable you to set up and configure various parameters in the MDO application. This chapter lists the settings for config.properties, suite.properties, and gdconfig.properties.

Although you can update or set the values for the properties in these configuration files to configure the MDO application, Oracle recommends that you make a copy of the file in the retailer sub-folder, and then update the configuration file. The properties defined in the configuration file (in the retailer sub-folder) override those found in the original configuration file.

Once you update the value of any parameter in the configuration files, you must restart the application server for the changes to take effect.

config.properties Settings

The following settings are contained in config.properties. Each property is shown with its default value.

Table 12–1 Settings for config.properties

Property	Description
p4pgui.what-if.max.size=1000	The maximum weighted size (hard limit) of a What-If selection. This value should not be set to greater than 1,000. The What-If recalculation will not be initiated if the number of items exceeds this value. Use this parameter to limit the size of a user's What If recalculation so that overall performance is acceptable.
p4pgui.what-if.warn.size=100	The practical maximum weighted size (soft limit) of a What-If selection. This value is best if set to 100. Setting this to a higher value can impact performance. The What-If recalculation can still be initiated if the number of items exceeds this value; however, the user will receive a warning. Use this parameter to warn users of particularly expensive recalculations.
p4pgui.what-if.pricing-group-item.weight=0.7	The weight to use for each item after the first in a pricing group in a What-If recalculation. Use this value as a variable when recalculating items in a pricing group. The value reflects the relative cost of optimizing individual items as compared to optimizing items in a pricing group.
pricefe.systemwide.itemDominance=true	Flag for setting pricing group dominance or item dominance at the system level. This includes OTB and What If.
pricefe.otb.enabled=true	Flag for exposing the OTB functionality in the UI.
pricefe.otb.excludeDCInventory=false	Flag for setting how the markdown budget is calculated. If set to true, then the calculation will exclude the markdown costs of the distribution center inventory.
pricefe.seasoncodes.error.limit=100	The maximum weighted size (hard limit) of a Seasonality Curve selection. This value should not be set to greater than 100. The Seasonality Curve selection will not be initiated if the number of curves exceeds this value.
pricefe.seasoncodes.warn.limit=20	The practical maximum weighted size (soft limit) of a Seasonality Curve selection. The recommended value is 20. Setting this to a higher value can impact performance. The Seasonality Curve selection can still be initiated if the number of curves exceeds this value; however, the user will receive a warning.
p4pgui.whatif.fcstSalesUnitsSeries.color = #87CEEB	Sets the color for the forecast (model run) sales units curve.
p4pgui.whatif.currSalesUnitsSeries.color = #4169E1	Sets the color for the current (recalc) sales units curve.
p4pgui.whatif.fcstInvSeries.color = #FF6347	Sets the color for the forecast (model run) inventory curve.
p4pgui.whatif.currInvSeries.color = #FF0000	Sets the color for the current (recalc) inventory curve.
p4pgui.whatif.fcstTicketPriceSeries.color = #CD5C5C	Sets the color for the forecast (model run) ticket price curve.
p4pgui.whatif.currTicketPriceSeries.color = #8B0000	Sets the color for the current (recalc) ticket price curve.
p4pgui.whatif.fcstSalesPriceSeries.color = #90EE90	Sets the color for the forecast (model run) sales price curve.
p4pgui.whatif.currSalesPriceSeries.color = #008000	Sets the color for the current (recalc) sales price curve.
p4pgui.whatif.fcstBaseSalesSeries.color = #FFD700	Sets the color for the forecast (model run) base sales (de_promo_de_price) curve.

Table 12–1 (Cont.) Settings for config.properties

Property	Description
p4pgui.whatif.currBaseSalesSeries.color = #DAA520	Sets the color for the current (recalc) base sales (de_promo_de_price) curve.
p4pgui.whatif.promotions.color = #FFE4B5	Sets the color for the promotions.
pricefe.itemListFilter.delimiters=newline, tab, space, colon, semicolon, comma	Identifies the delimiters used to separate the hierarchy list. Remove any delimiters if they occur in the identifier that is being filtered.
pricefe.itemListFilter.identifier=INT_UNIQUE_ID	In conjunction with the INT_UNIQUE_ID derivation in p4p-column-list.xml file, this is used to define the hierarchy combination for filtering.
pricefe.showItemLevelFilter.WarningMsg=false	Use to obtain more details on which items are invalid or hidden in item level filtering. When this is set to true, Item Level Find by Filter is implemented, and the hidden item warning message and the invalid item (or hierarchy) warning message are displayed. By default, this is set to false and the hierarchy level Find by Filter is implemented.
pricefe.showRegionFilter=true	Determines whether or not the region filter in the quick filter is displayed.
p4pgui.reports.concurrency = true	When this parameter is set to false, MDO processes one report at a time sequentially. When this parameter is set to true, MDO processes the reports requests in parallel mode.
p4pgui.max.reports = 25	This parameter defines the hard limit for the number of pending report requests. Users requesting reports will be prompted to try again later when the number of pending requests reaches this limit.

suite.properties Settings

The following settings are contained in suite.properties. Each property is shown with its default value.

Table 12–2 Parameters in the suite.properties File

Parameter	Description
common.dbdialect.dialect	Use this parameter to specify the database dialect used within the suite.
usermanagement.login.url	Use this parameter to specify the User Management login URL.
usermanagement.manageUsers.url	Use this parameter to specify the URL for the Manage Users screen in the User Management utility.
usermanagement.changePassword.url	Use this parameter to specify the URL for the Change Password screen in the User Management utility.
businessrulemgr.entry.url	Use this parameter to specify the Business Rule Manager URL.
storesets.entry.url	Use this parameter to specify the Store Set Management URL.
p4pgui.login.url	The MDO application login URL.
common.hierarchy.cache.timeout.hours	Number of hours for the hierarchy caches to become stale.

Table 12–2 (Cont.) Parameters in the suite.properties File

Parameter	Description
common.hierarchy.fetch.merch.maxlevels	Maximum number of merchandise hierarchy levels to fetch at a time.
common.hierarchy.fetch.loc.maxlevels	Maximum number of location hierarchy levels to fetch at a time.
common.hierarchy.merch.chainid	Identification number of the merchandise hierarchy chain.
common.hierarchy.loc.chainid	Identification number of the location hierarchy chain.
common.jdbc.oracle.fetchsize	The JDBC fetch size for result set on a Oracle database.
common.jdbc.db2.fetchsize	The JDBC fetch size for result set on a DB2 database.
common.dump.csv.forecast.response	Use this parameter to specify that *.csv* files are created for forecast response.
common.help.columnDef	Use this parameter to specify the HTML online help file that contains the column definitions for context-sensitivity.
common.help.customizeTable	Use this parameter to specify the HTML online help file that contains the customized table definitions for context sensitivity.
common.help.printExport	Use this parameter to specify the HTML online help file that appears when an user chooses to print or export information on the user interface.
delphi.rmi.host	Use this parameter to specify the Delphi URL for interactive Calculation Engine use.
delphi.rmi.port	Use this parameter to specify the Delphi port for interactive Calculation Engine use.
suite.loginform.autocomplete	Use this parameter to use the AutoComplete feature in the User Management utility.
suite.httpsession.timeout	Use this parameter to specify the duration, in seconds, for the HTTP session time out. This parameter applies across the suite.
suite.userlogin.timeout	Use this parameter to specify the duration, in seconds, for the user login time out. This parameter applies across the suite.
suite.cookie.secure	Use this parameter to specify a secure cookie. This parameter applies across the suite.
suite.cookie.domain	Defines the domain where the SSO cookie is. If left empty, then the default value is used.
suite.logoutpage.show	Used when MDO is integrated with Oracle SSO. Either shows successful logout page or redirects to login page. Default is false.
suite.logintimeout.manage	Use this parameter to manage login time outs. The value defaults to *Fault*, and indicates the login time out defaults to session time out.
common.spread.fontname	Use this parameter to specify the font used in the Spread feature.

Table 12–2 (Cont.) Parameters in the suite.properties File

Parameter	Description
hierarchy.displayType	Use this parameter to set the hierarchy display type that displays in the hierarchy control. The acceptable values are ID, DESC, ID-DESC, and DESC-ID.
audit.groupname.excluded	Use this parameter to specify the list of audit event groups that will not be logged. The value defaults to USER_GROUP, and indicates that all User Management events are excluded. To log auditing, leave the value blank. A value of MDO_WS_GROUP turns off auditing of the worksheet status. A value of COS_GROUP turns off auditing of the remote user. For Grid Designer: excluding GD_GROUP will turn off auditing for GD_SAVE, GD_PUBLISH, GD_BACKUP actions.
copyright.date	Used by Installer to resolve the copyright date.

gdconfig.properties Settings

The following settings are contained in gdconfig.properties.These properties are used with the Grid Designer.

Table 12–3 Parameters in the gdconfig.properties File

Parameter	Description
gd.resource.supported.locales	Lists the locales supported by the application. it must only be updated when new supported languages are added to the product.
gd.resource.configured.locales	Sets the default locale used in the Resource Browser Editor.
gd.publish.conf.host.names	Lets you set up a list of host names in a cluster where you want to publish the configuration. For standalone applications, leave this value blank.

13

Standard Reports

The MDO application comes with a group of default reports.

This chapter contains the following sections:

Introduction

Markdown Optimization provides standard reports, which users generate and view from the application UI. These standard reports and plug-in custom reports can be configured using XML.

The generated reports are presented to the user as Excel spreadsheets. Basic formatting of the spreadsheets is defined in the XML configuration file. for complex formatting, use VBA macros in Excel spreadsheet templates.

Development and maintenance of standard reports is easier using XML. A set of report requirements can be satisfied through the use of standard reports if the following qualifications are met:

- the generated report must include a single table of results that is comprised of rows and columns
- all of the required report data can be selected directly or derived from data contained in the p4p_display_items database view
- the aggregated data is hierarchical in nature
- the data for the report can be filtered by the selection of one or more Markdown Optimization worksheets

  Or

- the generated report can be developed using VBA macros that manipulate data in a spreadsheet that satisfies the above requirements

The Configuration Process for Standard Reports

The Markdown Optimization standard reporting infrastructure is based on a standard report generator Java class (GenericP4PItemReportGenerator) and a standard report filter Java class (GenericP4PItemReportFilter). Report XML files are identified in the config.properties files and located in the grids directory in the application configuration directory structure. Each report XML file configures a single standard

report. The report XML files use string resources that are defined in the gridResources.properties file.

The Config.Properties File Setup

The config.properties file is used to identify the standard reports to be configured using XML. The reportKeys property is a comma-separated list of the properties that are used to specify the name of the XML file for the report. Here is a sample setup of the config.properties file:

```
# Reports
reportKeys=sample-plugin-report,sample-md-analysis-report-1,sample-price-change-
report-1
sample-plugin-report=sample-plugin-report.xml
sample-md-analysis-report-1=sample-md-analysis-report-1.xml
sample-price-change-report-1=sample-price-change-report-1.xml
```

The Report XML Structure

The report XML structure is based on the grid XML structure in Markdown Optimization:

report

- worksheet-filter
- page-setup
- column-group-spec
  - column
    - key
    - parent-key
    - column-properties
    - custom-property
- row-group
  - key
  - rowgroup-properties
  - column-group-override
    - column
      - key
      - parent-key
      - column-properties
      - custom-property
- row-group

Report Element Definitions

The report XML structure contains the following elements:

- report: the root element in the XML file. It includes the following attributes:

- name: the resource string that indicates the name of the report (shown in the report list displayed in the UI).
- generator-class: the Java class used to generate the report. For standard reports, this is com.profitlogic.p4pgui.reports.appcommon.GenericP4PItemReportGenerator.
- show-row-group-only: used ti indicate that the report does not show any detail lines, only row groups. Values are true or false.
- show-report-header: used to indicate if the report header is shown. Values are true and false.
- where-clause: the SQL used to filter data for the result set.
- order-by-clause: the SQL used to establish the order of data in the result set. This must include the order of aggregations.
- template: the name of the Excel spreadsheet used as a template for the report output.
- report-group: the resource string that indicates the name of the report group. The report-group defines the reporting tabs in the application UI.

- worksheet-filter: used to configure parts of the filter that are displayed before a user runs a report. It includes the following attributes:
  - filter-class: the Java class used to render the filters UI. For standard reports, this is com.profitlogic.p4pgui.reports.uicommon.GenericP4PItemReportFilter.
  - subtotals-check box: indicates if a check box to allow the user to enable/disable subtotals is shown in the UI.
  - select: the SQL statement that indicates which worksheets to select for filter display.
  - where-clause: the SQL used to filter which worksheets are displayed in the filter.
  - order-by-clause: the SQL used to order the worksheets in the filter.
  - label: the resource string for the worksheet filter label.
  - allow-all-worksheets - indicates if all worksheets or only those to which a user has view access are available.
- page-setup: used to configure the page setup options for the output Excel report. It includes the following attributes:
  - orientation: portrait or landscape.
  - fit-to-page-height: the fit-to-page-height option in Excel.
  - fit-to-page-width: the fit-to-page-width option in Excel.
  - page-size: letter, legal, A4, or A5.

The other elements used in the report XML are the same as those used in the grid XML, which is described in the grid configuration documentation.

Report XML Validations and Rules

These rules and validations must be followed when configuring the standard report using the report XML:

- Only one column-group-spec is allowed in the report XML. The single column-group-spec includes the list of columns that are shown in the report (as long as the column is visible).
- A custom-property called "style" is required for every report column. The value of the style property must be one of the legal column styles for the reports.

  <custom-property name="style" value="STRING" custom-type="application"/>
- The report element can have at most one child row-group element, and each row-group can contain at most one child row-group. This allows the aggregations in the report to be shown in a hierarchical fashion.
- The row-group group by element must refer to a column key that is included in the report column list.
- Only columns defined in the override-column-group are shown in a row-group (aggregation).
- Each column in the override-column-group in a row-group must also be a column in the report column-group-spec that is visible. The reference is done using the column key child element.
- The legal functions available for columns in the override-column-group in the row-group are listed, along with their behaviors.
- The column type cannot be overridden in a column in an override-column-group. The type is assumed to be the same as the type identified in the column in the report column list.
- The following properties included in the column-properties element are ignored by reports: description, resource, template, display-type, read-only-type, format, db-table-name, groupId, group-description, filterable, sortable, orderable, hideable, expandable, editable, editable-in-readmode, composeable, operatortype, filter-enum-sql, and columntype.
- The following properties are the only ones honored in an override column specified in a row-group: db-column-name, and visibility. All other properties are taken from the column as defined in the report column list.
- Show Row Group Only / Subtotals Checkbox: If the show-row-group-only option is set to true, then the subtotals check box option must be false.

Additional Guidelines

Here are some additional guidelines to consider when configuring the report XML.

Weighted Averages

If a report needs to show a unit cost weighted average based on inventory units and unit cost, use the following weighted average formula:

S (inv units * unit cost) / S (inv units)

The column used to display the weighted average is the unit cost column. A invisible derived column must be created that is defined as the unit cost multiplied by the inventory units. The row grouping that displays the weighted average must use the P4P_DIVIDE function specifying the derived column name as the first argument and the inventory units column as the second argument. For example:

<column-properties>

  <function key="P4P_DIVIDE">

```
    <args>total_cost</args>
    <args>unit_cost</args>
  </function>
</column-properties>
```

Aggregation Rows

This section describes how to show an aggregation row that is neither the value of the aggregated string nor a value that can be read from a resource file.

A report is required to aggregate the initial and subsequent recommendations. The department header must show whether it is the initial grouping or the subsequent grouping. The report includes a derived column that prepares the initial/subsequent grouping string. The derived column is not visible in the standard column list, but is used in the row group aggregation for department. For example, the derived column in the column list could have the following derivation:

derivation="(case when markdown_number < 2 then 'Initial Markdowns' else 'Further Markdowns' end)"

In the row grouping, the hierarchy column that displays the derived column data overrides the db-column-name to show the derived column instead of the standard column data:

```
<column>
        <key>HIERARCHY3</key>
        <parent-key/>
        <column-properties db-column-name="derived_column_name">
                <function key="P4P_MULTI_STRING"/>
        </column-properties>
        <custom-property name="style" value="STRING" custom-type="application"/>
</column>
```

Custom Plug-In Report XML

Custom reports that are developed using Java and plugged into Markdown Optimization also require an XML specification so that they are registered with the application. The XML required is a subset of the XML required for specifying standard reports. The XML must include the report and worksheet-filter elements.

Note that if the custom report is developed to use other configuration information from the XML file, other elements may be required for that specific custom report.

Here is sample custom plug-in report

```
XML:<?xml version="1.0" encoding="UTF-8"?>
<report
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
        xsi:noNamespaceSchemaLocation="Grid.xsd"
        name="p4pgui.reports.testPluginReport"
        generator-class="com.profitlogic.p4pgui.reports.test.TestReport"
        report-group="p4pgui.reports.sampleGroupName">
        <worksheet-filter
           filter-class="com.profitlogic.p4pgui.reports.test.TestReportFilter"/>
</report>
```

This report configuration registers a custom report with the application that uses the specified Java generator and filter classes. The report name and group specifications are resource strings that refer to the gridResources.properties file.

Additional Information

Here is some additional reference information.

Valid Formats

Table 13–1 Valid Formats for Markdown Optimization Standard Reports

Entity	description	Example
Strings		
STRING	A left-aligned string	SAMPLE
CENTER_STRING	A centered string	SAMPLE
Numeric Data Types		
CURRENCY	A numeric column shown with a $ and two decimal places	$34.23
CURRENCY_ONE_DP	A numeric column shown with a $ and one decimal place	$34.2
WHOLECURRENCY	A numeric column shown with a $ and no decimal places	$34
PERCENT	A numeric column shown with a % symbol, multiplied by 100 and with two decimal places	34.23%
WHOLEPERCENT	A numeric column shown with the % symbol, multiplied by 100 and with no decimal places	34%
NUMBER	A numeric column shown with two decimal places	34.23
NUMBER_ONE_DP	A numeric column shown with one decimal place	34.2
WHOLENUMBER	A numeric column shown with no decimal places	34
Dates		
DATE_MM_DD_YYYY	A date format following mm/dd/yyyy	10/23/2005
DATE_MM_DD	A date format following mm/dd	10/23
DATE_DD_MMM_YY	A date format following dd-mmm-yy	23-Oct-05

Available Column Functions Valid functions for numeric columns include:

- P4P_SUM – the sum of all columns.
- P4P_MIN – the minimum numeric value
- P4P_MAX – the maximum numeric value.
- P4P_AVG – the average of all numeric values.
- P4P_DIVIDE – requires two argument children elements. The arguments must be references to columns defined in either the override column list of the row group or in the top level column list. If the columns are specified in the override column list of the row group, then the divide function divides by how the column functions are specified in that row group. For example, if the first column is specified as a SUM and the other is specified as a MIN, then the divide function divides the SUM of the first by the MIN of the second. If either of the columns are not specified in the override column list, the column value used is the sum.

Valid functions for date columns include:

- P4P_MIN – the minimum date.
- P4P_MAX – the maximum date.

Valid functions for string columns are:

- P4P_MULTI_STRING – used to show the string itself if it is the same for all detail records within a column group, blank, or a resource string identified by the arguments element, and if multiple values exist within the group row.
- P4P_STRING_COUNT – the total number of unique strings.

Functions Summary

Table 13–2 Standard Report Functions Summary

Function	Available for Numeric Data Type	Available for Date Data Type	Available for String Data Type	Arguments
P4P_SUM	X	NA	NA	No
P4P_MIN	X	X	NA	No
P4P_MAX	X	X	NA	No
P4P_AVG	X	NA	NA	No
P4P_DIVIDE	X	NA	NA	Two required
P4P_MULTI_STRING	NA	NA	X	One optional
P4P_STRING_COUNT	NA	NA	X	No

Available Paper Sizes The valid page sizes available for use in the page-setup element include Letter, Legal, A4, and A5.

Limitations on Reports

Consider the following application-specific guidelines:

- No limitations exist on the number of users who can access reports.
- No limitations exist on the size of the spreadsheet.
- Large reports may take some time to download.
- Generating reports is a memory intensive operation. The amount of memory actually required depends on the amount of underlying report data and the complexity of the report. It is indeed possible to consume all available server memory or severely compromise performance of the application if you are running too many reports simultaneously or generating large reports.
- Reports are generated using SQL queries to fetch report data and process the data further in java code. So the performance for generating reports is also tied to performance of database queries and the java code that generates reports.

Report Generator

Note the following:

A new report generator class called com.profitlogic.p4pgui.reports.appcommon.GenericFlatItemReportGenerator should

replace the standard class in the grid configuration file for reports. This file differs from the standard class in that the new class does not support hierarchical reports, and hence it does not use the accumulators to stage the data. Instead, it feeds directly into POI for generating the Excel stream. The name of the class should be set as the value for the generator-class attribute of the report element. This cuts down the memory usage by a third.

A per-report optionally configurable throttle should be added to the reports. An attribute max-rows can be specified in the report element. The number that is set for this attribute will represent the maximum number of rows returned in the report. If the report SQL gives more rows, the first max-rows number of rows will be returned.

A per-report optionally configurable attribute should be added to the reports. An attribute zip-stream can be specified in the report element and can have value of either false (default) or true. When true, the report stream will be zipped before persisting to the database. Empirical tests have shown reports having rows > 20,000 to be ~= 32 MB, which is the size of the database BLOB field.

```
The JVM settings for JRockit must include the following:
-Xgc:gencon -Xgcpause -Xns=<1/5th to 1/4th MAX_HEAP>m -Xms<MIN_HEAP>m -Xmx<MAX_
HEAP>m
```

14

RDM Data Mapping

This chapter addresses RDM data.

It contains the following sections:

Overview of RDM Data Mapping

The Retail Data Mart (RDM) abstracts forecasting and historic data from the base applications for use with Business Intelligence tools.

The following are RDM concepts:

- Fact

  A fact is a discrete item of business information. Facts are typed as descriptive or metric.

- Metric

  Metric refers to a piece of measurable data, which is a derivative of a quantifiable fact. Metrics capture quantifiable business facts that may be used as business measures - any data that may be mathematically manipulated to produce meaningful information.

- Attribute

  An attribute is a property or characteristic of a dimension that may be stored as a data fact. Attributes represent descriptive elements of a certain level of the dimension hierarchy.

- Dimension

  A dimension is an aspect or perspective by which the facts or metrics may be accessed, selected, sequenced, grouped, filtered, and aggregated. Each dimension consists of multiple dimension levels. A dimension is typically a text value, such as a region or a department, or has a date value.

RDM Facts

In order to obtain facts, the RDM uses data located in the following schemas:

CDW Schema

P4P Schema

PMA Schema

To enable analysis of the various measures and metrics at different levels of the hierarchy and to enhance performance, Roll-ups or Summary Tables may be needed at these Levels.

RDM Tables Mapped to Markdown Optimization Tables

The following table shows the mapping between the base application tables and RDM tables.

Table 14–1 RDM Tables Mapped to Markdown Optimization Tables

RDM Table	RDM Table Details	Related Markdown Optimization Table
RDM_MERCHANDISE_TBL	Reflects the current state of the corresponding application table, which is loaded weekly.	MERCHANDISE_TBL
RDM_LOCATION_TBL	Reflects the current state of the corresponding application table, which is loaded weekly.	LOCATION_TBL
RDM_PERIODS_TBL	Part of initial setup. Refreshed weekly. This table contains a client's fiscal calendar.	PERIODS_TBL
RDM_HIERARCHY_LEVELS_TBL	Reflects the current state of the corresponding application table, which is loaded once when the application is initially configured.	HIERARCHY_LEVELS_TBL
RDM_ITEMS_TBL	Reflects the current state of the corresponding application table, which is loaded weekly.	ITEMS_TBL
RDM_BUDGETS	Refreshed weekly to reflect corresponding application changes.	P4P_BUDGETS
RDM_ACTIVITIES	Refreshed weekly to reflect corresponding application changes. This table stores sales and inventory data.	ACTIVITIES
RDM_FORECAST_ACTIVITIES	Part of initial setup. Refreshed weekly.	FORECAST_ACTIVITIES
RDM_HIST_MARKDOWNS	Reflects the current state of the corresponding application table, which is loaded weekly.	HIST_MARKDOWNS_TBL

Table 14–1 (Cont.) RDM Tables Mapped to Markdown Optimization Tables

RDM Table	RDM Table Details	Related Markdown Optimization Table
RDM_ITEM_DATA	Part of initial setup. Refreshed during incremental load.	ITEM_DATA
RDM_MV_ACT_n (Optional summary table)	Refreshed at the time of the mview install.	MERCHANDISE_TBL, LOCATION_TBL, ACTIVITIES
RDM_MV_FA_n (Optional summary table)	Refreshed at the time of the mview install.	MERCHANDISE_TBL, LOCATION_TBL, FORECAST_ ACTIVITIES

RDM Data Mapped to Markdown Optimization Data

This section shows how RDM columns map to Markdown Optimization data.

RDM Item Data

The following table shows how the Retail Data Mart derives data from the application data. All of these columns are refreshed immediately following the weekly load. Some columns are also refreshed immediately following the incremental run, as noted in the table below.

Table 14–2 RDM Item Data Mapping

RDM_ITEM_DATA	Description	ITEM_DATA (ID)
ACCEPTED_MARKDOWN	This column is refreshed after both the incremental and weekly loads.	CASE WHEN (ID.recommended_retail_price IS NOT NULL AND ID.TAKEN_ PRICE IS NOT NULL AND ID.recommended_retail_price = ID.TAKEN_PRICE) THEN 1 ELSE 0 END
ACCEPTED_PRICE	Accepted mark down price. This column is refreshed after both the incremental load and the weekly load.	ID.TAKEN_PRICE
ADDED_MD_FLAG	Shows for a taken markdown on a non-recommended item, else 0. Summary rows show the total number of taken, non-recommended items underneath.	CASE WHEN (ID.recommended_retail_price IS NULL AND ID.collection_ recommended_price IS NULL) THEN 1 ELSE 0 END
AVG_PRICE	Average price of an item	ID.average_price
AVG_PRICE_LTD	The average unit retail price of the item life to date.	ID.std_average_price
AVG_PRICE_LW	The average unit retail price of the item last week.	CASE WHEN (ID.unit_sales_ through_week = 0) THEN 0 ELSE (ID.dollar_sales_through_ week / ID.unit_sales_through_ week) END
CHAIN_MAX_PRICE	The maximum price at the chain level.	rtrp.current_retail_price_max
CHAIN_MIN_PRICE	The minimum price at the chain level.	rtrp.current_retail_price_min

Table 14–2 (Cont.) RDM Item Data Mapping

RDM_ITEM_DATA	Description	ITEM_DATA (ID)
COL_OPPORTUNITY_COST	The opportunity cost (margin loss) of deferring taking a recommended markdown until the next available markdown date. Assumes that the deferred markdown will then be taken as recommended by the application.	ID.collection_opportunity_cost
COMMITTED_INV_UNITS	Committed inventory units	ID.committed_inv_units
CURRENT_RTL_PRICE	Current retail price as of last week's sales history.	ID.current_retail_price
CURRTL_PERC_OFF_ORRTL	Current retail price as a percentage off original retail price	ID.current_percent_off
DC_OH_UNITS	Distribution center on hand item units	ID.warehouse_on_hand
DC_OO_UNITS	Distribution center on order item units	ID.warehouse_on_order
DELAYED_MARKDOWN	Number of recommended markdowns not taken before sendback. This column is refreshed after both the incremental load and the weekly load.	CASE WHEN (ID.recommended_retail_price IS NOT NULL AND ID.TAKEN_ PRICE IS NULL AND ID.markdown_flag = 0) THEN 1 ELSE 0 END
EFFECTIVE_DATE	Date on which a MD taken on the item will be effective in stores.	ID.effective_date
ENDING_INV_UNITS	Projected inventory units on hand at the client specified out date.	ID.ENDING_INVENTORY_ UNITS
FA_INVENTORY_UNITS_NW	Forecasted inventory units NW	rfa2.inventory_units
FA_INVENTORY_UNITS_TW	Forecasted inventory units TW	rfa1.inventory_units
FA_SALES_DOLLARS_NW	Forecasted sales dollars NW	rfa2.inventory_units
FA_SALES_DOLLARS_TW	Forecasted sales dollars TW	rfa1.sales_dollars
FA_SALES_UNITS_NW	Forecasted sales units NW	rfa2.sales_units
FA_SALES_UNITS_TW	Forecasted sales units TW	rfa1.sales_units
FA_TICKET_PRICE_NW	Forecasted ticket price NW	rfa2.ticket_price
FA_TICKET_PRICE_TW	Forecasted ticket price TW	rfa1.ticket_price
FIRST_RECEIPT_DATE	First item receipt date	ID.FIRST_RECEIPT_DATE
FIRST_SALE_DATE	Date of first retail sale	fsd.first_sale_dt

Table 14–2 (Cont.) RDM Item Data Mapping

RDM_ITEM_DATA	Description	ITEM_DATA (ID)
FTB_RATIO	First Time Buyers ratio	CASE WHEN (CASE WHEN (ID.recommended_retail_price IS NOT NULL OR ID.collection_ recommended_price IS NOT NULL) THEN ID.committed_ inv_units * (ID.current_retail_ price- ID.recommended_retail_ price) ELSE 0 END) = 0 THEN 0 ELSE (ID.opportunity_cost / (CASE WHEN (ID.recommended_retail_price IS NOT NULL OR ID.collection_ recommended_price IS NOT NULL) THEN ID.committed_ inv_units * (ID.current_retail_ price - ID.recommended_retail_ price) ELSE 0 END)) END
GROSS_PROFIT_AMT	Gross Margin money amount	NULL
GROSS_PROFIT_AMT_LTD	Life to Date gross margin money	ID.cumm_gross_profit_dollar
GROSS_PROFIT_PERC_LTD	Life to Date gross margin percentage.	ID.cumm_gross_profit_perc
GROSS_PROFIT_PERCENT	Gross Margin percent	NULL
INV_COST_AMT_OH	Cost of the inventory on hand in the stores.	(ID.current_units_on_hand * ID.unit_cost)
INV_COST_AMT_OO	Cost of the inventory on order in the stores.	(ID.current_units_on_order * ID.unit_cost)
INV_RTL_AMT_OH	Retail value of the inventory on hand in the stores.	ID.current_units_on_hand * ID.current_retail_price
INV_RTL_AMT_OO	Retail value of the inventory on order in the stores.	ID.current_units_on_order * ID.current_retail_price
INV_UNITS_OH	Inventory on hand in the stores as of the beginning of this week (end of last week)	ID.current_units_on_hand
INV_UNITS_OO	Inventory on order in the stores as of the beginning of this week (end of last week)	ID.current_units_on_order
INV_UNITS_WM1	Inventory on hand in the stores as of the end last week minus one week.	ID.week_minus_1_units_on_ hand
INV_UNITS_WM2	Inventory on hand in the stores as of the end last week minus two weeks.	ID.week_minus_2_units_on_ hand
INV_UNITS_WM3	Inventory on hand in the stores as of the end last week minus three weeks.	ID.week_minus_3_units_on_ hand
IS_FIRST_MD	1/0 flag if first markdown - 1	CASE WHEN (ID.markdown_ number = 1) THEN 1 ELSE 0 END
LAST_MD_EFFECTIVE_DT	The date of the last markdown	hm.caldt
LAST_RECEIPT_DATE	Last item receipt date	ID.LAST_RECEIPT_DATE
LAST_REFRESH_DATE	Last time item data is refreshed. This column is refreshed after both the incremental and weekly loads.	TRUNC (SYSDATE)
LOCATION_ID	The location identifier	ID.LOCATION_ID

Table 14–2 (Cont.) RDM Item Data Mapping

RDM_ITEM_DATA	Description	ITEM_DATA (ID)
LOWEST_PROMO_PRICE	Planned lowest promo price for the item, on-going or at any point in the future.	ID.lowest_future_promotes_ price
MARKUP_PERC	Initial markup percent is the percentage of the initial retail price that is markup above cost.	ID.markup_percent
MD_AMT	Markdown amount	(ID.current_units_on_hand + ID.current_units_on_order - ID.weekly_projected_unit_sales) * (ID.current_retail_price - ID.recommended_retail_price)
MD_FLAG	Indicator for whether the item/pricing group received a markdown recommendation for the next effective date. This column is refreshed after both the incremental and weekly loads.	ID.MARKDOWN_FLAG
MD_NUMBER	The markdown number	ID.markdown_number
MD_TAKEN_THRU_OTB	Indicator for whether the item was assigned a markdown using the Optimize To Budget function.	CASE WHEN (ID.markdown_ flag = 4 AND w.worksheet_ status_id > 2) THEN 1 ELSE 0 END
MD_TYPE	The type of markdown	NULL -> daily MARKDOWN_ TYPE
MODIFIED_MARKDOWN	Number of modified markdowns from original recommendation. This column is refreshed after both the incremental and weekly loads.	CASE WHEN (ID.recommended_retail_price IS NOT NULL AND ID.TAKEN_ PRICE IS NOT NULL AND ID.recommended_retail_price <> ID.TAKEN_PRICE) THEN 1 ELSE 0 END
MODIFIED_OUT_OF_ STOCK_DATE	Modified target date by which the item should achieve its sell-through.	ID.out_of_stock_date
MODIFIED_TARGET_ST_ PERC	New target sell-through percentage.	NULL
NEXT_MD_DATE	Next recommended markdown date after the current markdown effective date.	ID.PROJECTED_NEXT_ MARKDOWN
NEXT_REC_PERC_OFF_ ORRTL	Recommended price as a percentage off original retail price	ID.NEXT_REC_PERC_OFF_ ORRTL
NEXT_RTL_PRICE	Next recommended markdown price after the current markdown effective date.	ID.NEXT_RTL_PRICE
NO_STORE_WITH_OH	Number of stores with inventory on hand at the end of last week.	ID.no_store_with_on_hand

Table 14–2 (Cont.) RDM Item Data Mapping

RDM_ITEM_DATA	Description	ITEM_DATA (ID)
NON_REC_MARKDOWN	Number of non recommended markdowns - calculated by looking at recommended item flag. This column is refreshed after both the incremental and weekly loads.	CASE WHEN (ID.recommended_retail_price IS NULL AND ID.TAKEN_PRICE IS NULL) THEN 0 WHEN (ID.recommended_retail_price IS NULL AND ID.TAKEN_PRICE IS NOT NULL) THEN 1 WHEN (ID.recommended_retail_price IS NOT NULL) THEN 0 END
O_USER_DATE_1 (1-6)	Custom metrics defined in inference rules.	ID.o_user_date_1 (1-6)
O_USER_FLOAT_1 (1-12)	Custom metrics defined in inference rules.	ID.o_user_float_1 (1-12)
O_USER_TEXT_1 (1-4)	Custom metrics defined in inference rules.	ID.o_user_text_1 (1-4)
OPPORTUNITY_COST	Margin loss. Opportunity cost of deferring taking a recommended markdown until the next available markdown date assuming that the deferred markdown will then be taken as recommended by the application.	ID.OPPORTUNITY_COST
ORIGINAL_EXIT_DATE	Original exit date setting for the item.	imt.out_dt
ORIGINAL_EXIT_DATE_ MOD_DT	Date/time when exit date was last changed.	NULL
ORIGINAL_RTL_PRICE	The retail price at the start of life of the item.	ID.original_retail_price
OUT_OF_STOCK_DATE	The target date by which the item should achieve its sell-through. This column is refreshed after both the incremental and weekly loads.	ID.OUT_OF_STOCK_DATE
OWNED_RTL_PRICE	The current “owned at” price for the item this week (including any pending markdowns expected to be effective this week).	ID.owned_rtl_price
PI_ID	Product ID.	ID.PI_ID
PLANNED_START_SELL_ DATE	Date on which an item is scheduled to begin activity.	ITEM_BRM_RULES.PLANNED_ START_DT
PRICE_LADDER_DESC	Label for the item's current price ladder.	l.ladder_name
PRICE_LADDER_TYPE	Indicator for whether the current price ladder is Price Point or Percent Off Ticket.	TO_CHAR (l.ladder_type)
PROJ_GM_AMT_EOL	The forecasted gross margin dollars for an item at the client specified inventory exit date.	ID.proj_std_eol_gm_amount
PROJ_GM_AMT_EOL_GRP	The forecasted gross margin dollars for an item at the client specified inventory exit date.	ID.proj_std_eol_gm_amount_c
PROJ_GM_PERC_EOL	The forecasted gross margin percent for an item at the client specified inventory exit date.	ID.proj_std_eol_gm_perc

Table 14–2 (Cont.) RDM Item Data Mapping

RDM_ITEM_DATA	Description	ITEM_DATA (ID)
PROJ_GM_PERC_EOL_GRP	The forecasted gross margin percent for an item at the client specified inventory exit date.	ID.proj_std_eol_gm_perc_c
PROJ_OH_COST_EOL	Projected cost of the inventory remaining at the out date.	ID.proj_oh_cost_eol
PROJ_OH_RTL_EOL	Projected inventory units on hand at the client specified out date.	ID.PROJ_OH_RTL_EOL
PROJ_OH_UNITS_EFF_DT	Expected number of units in stores when the markdown becomes effective.	ID.PROJ_OH_UNITS_EFF_DT
PROJ_OH_UNITS_EOL	Projected inventory units on hand at the client specified out date.	CASE WHEN (ID.original_ retail_price = 0) THEN 0 ELSE (ID.proj_oh_rtl_eol / ID.original_retail_price) END
PROJ_OUT_OF_STOCK	The projected date is either: – the date at which all inventory is projected to see through completely (i.e., 100% sellthru) or – if the model does not expect it to sell through completely, then the projected outdate becomes the week-ending Saturday of the outdate itself	ID.PROJECTED_OUT_OF_ STOCK
PROJ_RTL_PRICE_EOL	Projected retail price at the out date.	CASE WHEN (ID.ending_ inventory_units = 0) THEN 0 ELSE (ID.proj_oh_rtl_eol / ID.ending_inventory_units) END
PROJ_SALES_AMT_EOL	Projected total sales dollars for an item at the end of its life, assuming all application recommendations are taken.	ID.EOL_CUM_DOLLARS_ SALES
PROJ_SALES_UNITS_EOL	Projected total sales units for an item at the end of its life, assuming all application recommendations are taken.	ID.EOL_CUM_UNIT_ SALES+id.CUMULATIVE_ QUANTITY_SOLD
PROJ_ST_PERC_EOL	Projected percent of total inventory quantity sold by the client specified out date, assuming all application recommendations are taken.	NULL
PROJ_UNITS_OH_NW	Projected unit inventory of recommended markdowns for markdown week.	(ID.current_units_on_hand + ID.current_units_on_order - ID.weekly_projected_unit_sales)
PROMO_DESC	This is the client's text description of a planned promotional event.	pp.promo_desc
PROMO_END_DT	The date the promotion ends.	pp.end_dt
PROMO_FLAG	Identifies items that have a planned promotion this week or in the future.	ID.promotion_flag
PROMO_PCT_OFF	The value for the percentage off promotion	pp.promo_pct_off
PROMO_PRICE	The value for the promotion price.	pp.promo_price

Table 14–2 (Cont.) RDM Item Data Mapping

RDM_ITEM_DATA	Description	ITEM_DATA (ID)
PROMO_START_DT	the date the promotion starts.	pp.start_dt
REC_AS_COLLECTION	Shows 1 for a recommended pricing group, 0 for a non-recommended pricing group. Summary rows show the total number of recommended pricing groups underneath.	CASE WHEN (ID.collection_ recommended_price IS NOT NULL) THEN 1 ELSE 0 END
REC_AS_ITEM	Shows 1 for a recommended item, 0 for a non-recommended item. Summary rows show the total number of recommended items underneath.	CASE WHEN (ID.recommended_retail_price IS NOT NULL) THEN 1 ELSE 0 END
REC_COLLECTION_PRICE	Recommended markdown price for a pricing group.	ID.collection_recommended_ price
REC_ITEM_FLAG	Indicator for whether the item/pricing group received a markdown recommendation for the next effective date. This column is refreshed after both the incremental and weekly loads.	ID.RECOMMENDED_ITEM_ FLAG
REC_MD_AMT	Markdown cost (using retail accounting) of taking the recommended markdown.	CASE WHEN (ID.recommended_retail_price IS NOT NULL OR ID.collection_ recommended_price IS NOT NULL) THEN ID.committed_ inv_units * (ID.current_retail_ price - ID.recommended_retail_ price) ELSE 0 END
REC_MD_INV_COST	Cost of inventory that is recommended for markdown.	ID.COMMITTED_INV_UNITS * ID.UNIT_COST
REC_PERC_OFF_CURTL	Recommended markdown price as a percentage off of current retail.	CASE WHEN ID.current_retail_ price = 0 THEN 0 ELSE (1 - ((CASE WHEN ID.process_as_ item = 1 THEN ID.recommended_retail_price ELSE ID.collection_ recommended_price END) / ID.current_retail_price)) END
REC_PERC_OFF_ORRTL	Recommended markdown price as a percentage off of original retail.	CASE WHEN ID.original_retail_ price = 0 THEN 0 ELSE (1 - ((CASE WHEN ID.process_as_ item = 1 THEN ID.recommended_retail_price ELSE ID.collection_ recommended_price END) / ID.original_retail_price)) END
REC_RTL_MAX	The maximum recommended retail price for the merchandise within a grouping.	ID.recommended_retail_price
REC_RTL_MIN	The minimum recommended retail price for the merchandise within a grouping.	ID.rec_rtl_min
REC_RTL_PRICE	Recommended markdown price for an item.	ID.RECOMMENDED_RETAIL_ PRICE

Table 14–2 (Cont.) RDM Item Data Mapping

RDM_ITEM_DATA	Description	ITEM_DATA (ID)
RECOMMENDED_ MARK-DOWN	1/0 recommended for markdown or not. This column is refreshed after both the incremental and the weekly loads.	CASE WHEN (ID.recommended_retail_price IS NOT NULL OR ID.collection_ recommended_price IS NOT NULL) THEN 1 ELSE 0 END
REVENUE_LOST_BUDGET_ CONST	Projected revenue difference between the recommended price and the OTB accepted price.	ID.revenue_lost_budget_const
SALES_AMT	Last week's sales dollars.	ID.dollar_sales_through_week
SALES_AMT_LTD	The total sales dollars since the item's Start Sell Date.	ID.cumulative_sales_dollars
SALES_AMT_WM1	The total dollar sales two weeks ago (last week minus one week)	ID.dollar_sales_week_minus_1
SALES_AMT_WM2	The total dollar sales three weeks ago (last week minus two weeks).	ID.dollar_sales_week_minus_2
SALES_AMT_WM3	The total dollar sales four weeks ago (last week minus three weeks).	ID.dollar_sales_week_minus_3
SALES_UNITS	The number of units sold for the week.	ID.unit_sales_through_week
SALES_UNITS_LTD	The total number of units sold since the item's Start Sell Date.	ID.cumulative_quantity_sold
SALES_UNITS_WM1	The number of units sold two weeks ago (last week minus one week).	ID.unit_sales_week_minus_1
SALES_UNITS_WM2	The number of units sold three weeks ago (last week minus two weeks).	ID.unit_sales_week_minus_2
SALES_UNITS_WM3	The number of units sold four weeks ago (last week minus three weeks).	ID.unit_sales_week_minus_3
SALVAGE_VALUE_AMT	The amount of the salvage value.	ID.salvage_value_perc * ID.proj_ oh_rtl_eol
SALVAGE_VALUE_PERC	The salvage value expressed as a percentage.	ID.salvage_value_perc
SELL_THROUGH	The number of units sold last week divided by the units on hand at the start of that week.	ID.sell_thrupercent_current_ week
SELL_THROUGH_LTD	This is the percent of store inventory sold from the beginning of the life up to the current date.	ID.cumulative_sellthru_percent
SELL_THROUGH_WM1	The number of units sold 2 weeks ago divided by the units on hand at the start of that week.	ID.sell_thrupercent_week_ minus_1
SELL_THROUGH_WM2	The number of units sold 3 weeks ago divided by the units on hand at the start of that week.	ID.sell_thrupercent_week_ minus_2
SELL_THROUGH_WM3	The number of units sold 4 weeks ago divided by the units on hand at the start of that week.	ID.sell_thrupercent_week_ minus_3
SENDBACK_DATE	Sendback date for the markdown.	ID.sendback_date
SENT_DATE	The date the markdown was sent to the price change system.	ID.sent_date

Table 14–2 (Cont.) RDM Item Data Mapping

RDM_ITEM_DATA	Description	ITEM_DATA (ID)
SENT_LADDER_ID	The price ladder that was sent to the price change system.	ID.sent_ladder_id
SENT_MARKDOWN_PRICE	The markdown price that was sent to the price change system.	ID.sent_markdown_price
START_SELL_DATE	Date on which an item is to begin activity.	ID.START_SELL_DATE
SUBMITTAL_WORKSHEET_ID	Key to P4P_SUBMITTAL_ WORKSHEET, linking item to worksheet.	ID.submittal_worksheet_id
TAKEN_DEEPER	1/0 per item indicating whether a recommended item was taken to a price at least 10% deeper than the recommended price. This column is refreshed after both the incremental and weekly loads.	CASE WHEN (ID.TAKEN_ PRICE < ID.recommended_ retail_price) THEN 1 ELSE 0 END
TAKEN_MARKDOWN	Markdown status of the item- Taken or Not Taken. This column is refreshed after both the incremental and weekly loads.	CASE WHEN (ID.TAKEN_ PRICE IS NOT NULL AND ID.markdown_flag <> 0) THEN 1 ELSE 0 END
TAKEN_MD_AMT	Taken markdown amount	CASE WHEN (ID.TAKEN_ PRICE IS NOT NULL AND ID.markdown_flag <> 0) THEN ID.committed_inv_units * (ID.current_retail_price - ID.TAKEN_PRICE) ELSE 0 END
TAKEN_MD_INV_COST	Inventory cost associated with markdowns taken.	(ID.committed_inv_units * ID.unit_cost)+((ID.current_ units_on_hand + ID.current_ units_on_order-ID.weekly_ projected_unit_sales)*ID.unit_ cost)
TAKEN_MD_PRICE	Taken markdown price	CASE WHEN (ID.TAKEN_ PRICE IS NOT NULL AND ID.markdown_flag <> 0) THEN TAKEN_PRICE ELSE 0 END
TAKEN_PERC_OFF_CURTL	Taken markdown price as a percentage off of current retail.	NULL
TAKEN_PERC_OFF_ORRTL	Taken markdown price as a percentage off of original retail.	ID.recommended_markdown_ perc
TAKEN_SHALLOWER	1/0 per item indicating whether a recommended item was taken to a price at least 10% higher than the recommended price. This column is refreshed after both the incremental and weekly loads.	CASE WHEN (ID.recommended_retail_price < ID.TAKEN_PRICE) THEN 1 ELSE 0 END
TARGET_INV_UNITS_OH_ EOL	Target inventory remaining.	imt.target_inventory_units
TARGET_ST_PERC	Target sell-through percentage considered by the last optimization.	1 - ID.ending_inventory_perc
TEMP_MD_FLAG	Indicates whether a taken markdown is permanent or temporary.	CASE WHEN (l.ladder_type > 1) THEN 1 ELSE 0 END
TOTAL_INV	Total inventory units	ID.committed_inv_units

Table 14–2 (Cont.) RDM Item Data Mapping

RDM_ITEM_DATA	Description	ITEM_DATA (ID)
UNIT_COST	Unit cost	ID.unit_cost
USER_DATE_1 (1-6)	Custom pass-through metrics.	ID.user_date_1 (1-6)
USER_FLOAT_1 (1-12)	Custom pass-through metrics.	ID.user_float_1 (1-12)
USER_TEXT_1 (1-4)	Custom pass-through metrics.	ID.user_text_1 (1-4)
WORKSHEET_STATUS_ID	The ID for the status of the submittal worksheet.	p4p_submittal_ worksheets.worksheet_status_id
WOS	Number of weeks of supply on hand at the start of this week.	ID.weeks_of_supply

RDM Item Synonyms

The RDM_ITEMS_TBL table contains Item synonyms pointing to Markdown Optimization.

Table 14–3 RDM Item Synonym Mapping

RDM_ITEMS_TBL	ITEMS_TBL
ITEM_ID	ITEM_ID
PI_ID	PI_ID
MERCHANDISE_ID	MERCHANDISE_ID
LOCATION_ID	LOCATION_ID
FULL_PRICE	FULL_PRICE
FIRST_RECEIPT_DT	FIRST_RECEIPT_DT
CLEARANCE_IND_DT	CLEARANCE_IND_DT
TARGET_INVENTORY_UNITS	TARGET_INVENTORY_UNITS
ITEM_STATUS	ITEM_STATUS
VENDOR_ID	VENDOR_ID
CLEARANCE_DT	CLEARANCE_DT
CURRENT_COST_AMT	CURRENT_COST_AMT
CURRENT_RETAIL_BEGIN_DT	CURRENT_RETAIL_BEGIN_DT
LAST_RECEIPT_DT	LAST_RECEIPT_DT
VENDOR	VENDOR
VENDOR_DESC	VENDOR_DESC
UNIT_COST	UNIT_COST
SEASON_CODE	SEASON_CODE
MODEL_START_DT	MODEL_START_DT

RDM Item CDA Data Views

The RDM_ITEMS_CDA_VW view maps to the PL_DD_ATTRIBUTES column where tablename='ITEMS_CDA_TBL' (ITEMS_CDA_TBL=ict, RDM_ITEMS_TBL=rit).

Table 14–4 RDM Item CDA View Data Mapping

RDM_ITEMS_CDA_VW	PL_DD_ATTRIBUTES
ITEM_ID	ict.ITEM_ID ITEM_ID
PI_ID	rit.PI_ID PI_ID
LOCATION_ID	rit.LOCATION_ID LOCATION_ID
ATTRIBUTE1_CH_ NAME (1-8)	(select attributename from PL_DD_ATTRIBUTES where tablename='ITEMS_CDA_TBL' and columnname='ATTRIBUTE1') ATTRIBUTE1_CH_ NAME
ATTRIBUTE1 (1-8)	ict.ATTRIBUTE1 ATTRIBUTE1
ATTRIBUTE1_CH_ ISDISABLED (1-8)	(select isdisabled from PL_DD_ATTRIBUTES where tablename='ITEMS_CDA_TBL' and columnname='ATTRIBUTE1') ATTRIBUTE1_CH_ ISDISABLED
ATTRIBUTE1_DT_ NAME (1-8)	(select attributename from PL_DD_ATTRIBUTES where tablename='ITEMS_CDA_TBL' and columnname='ATTRIBUTE1_DATE') ATTRIBUTE1_ DT_NAME
ATTRIBUTE1_DATE (1-8)	ict.ATTRIBUTE1_DATE ATTRIBUTE1_DATE
ATTRIBUTE1_DT_ ISDISABLED (1-8)	(select isdisabled from PL_DD_ATTRIBUTES where tablename='ITEMS_CDA_TBL' and columnname='ATTRIBUTE1_DATE') ATTRIBUTE1_ DT_ISDISABLED
ATTRIBUTE1_NU_ NAME (1-8)	(select attributename from PL_DD_ATTRIBUTES where tablename='ITEMS_CDA_TBL' and columnname='ATTRIBUTE1_NUMBER') ATTRIBUTE1_NU_NAME
ATTRIBUTE1_NUMBER (1-8)	ict.ATTRIBUTE1_NUMBER ATTRIBUTE1_NUMBER
ATTRIBUTE1_NU_ ISDISABLED (1-8)	(select isdisabled from PL_DD_ATTRIBUTES where tablename='ITEMS_CDA_TBL' and columnname='ATTRIBUTE1_NUMBER') ATTRIBUTE1_NU_ISDISABLED

RDM Activities Data

The RDM_ACTIVITIES table maps to the ACTIVITIES table as shown below.

Table 14–5 RDM Activities Data Mapping

RDM_ACTIVITIES	ACTIVITIES
ITEM_ID	ITEM_ID
CALENDAR_DT	CALENDAR_DT
PI_ID	PI_ID
LOCATION_ID	LOCATION_ID
PERIOD_ID	PERIOD_ID
NET_SALES_UNITS	NET_SALES_UNITS
NET_SALES_AMT	NET_SALES_AMT
GROSS_SALES_UNITS	GROSS_SALES_UNITS

Table 14–5 (Cont.) RDM Activities Data Mapping

RDM_ACTIVITIES	ACTIVITIES
GROSS_SALES_AMT	GROSS_SALES_AMT
POS_SALES_UNITS	POS_SALES_UNITS
POS_SALES_AMT	POS_SALES_AMT
FULL_SALES_UNITS	FULL_SALES_UNITS
FULL_SALES_AMT	FULL_SALES_AMT
CLR_SALES_UNITS	CLR_SALES_UNITS
CLR_SALES_AMT	CLR_SALES_AMT
RETURNED_UNITS	RETURNED_UNITS
FULL_PRICE	FULL_PRICE
CURRENT_RETAIL	CURRENT_RETAIL
CURRENT_INV_PRICE	CURRENT_INV_PRICE
TICKET_PRICE	TICKET_PRICE
SALES_PRICE	SALES_PRICE
NET_SALES_PRICE	NET_SALES_PRICE
GROSS_SALES_PRICE	GROSS_SALES_PRICE
ITEM_COST	ITEM_COST
INVENTORY_UNITS	INVENTORY_UNITS
DELIVERED_UNITS	DELIVERED_UNITS
ORDERED_UNITS	ORDERED_UNITS
SALVAGED_UNITS	SALVAGED_UNITS
BOH_UNITS	BOH_UNITS
BOH_AMT	BOH_AMT
STORE_COUNT	STORE_COUNT
STORE_COUNT_ON_ORDER	STORE_COUNT_ON_ORDER
STORE_COUNT_WITH_INV	STORE_COUNT_WITH_INV
COST_OF_SALES	NET_SALES_UNITS * items_tbl.UNIT_COST COST_OF_SALES

RDM Forecast Data

The RDM_FORECAST_ACTIVITIES contains only forecasts for future dates.

Table 14–6 RDM Forecast Data Mapping

RDM_FORECAST_ACTIVITIES	FORECAST_ACTIVITIES
PI_ID	PI_ID
LOCATION_ID	LOCATION_ID
PERIOD_ID	PERIOD_ID
SALES_UNITS	SALES_PRICE
TICKET_PRICE	TICKET_PRICE

Table 14–6 (Cont.) RDM Forecast Data Mapping

RDM_FORECAST_ACTIVITIES	FORECAST_ACTIVITIES
SALES_PRICE	SALES_PRICE
SALES_DOLLARS	SALES_UNITS * SALES_PRICE
COST_OF_SALES	SALES_UNITS * ITEMS_TBL.UNIT_COST
INVENTORY_UNITS	INVENTORY_UNITS
COLLECTION_SALES_UNITS	NULL
COLLECTION_TICKET_PRICE	NULL
COLLECTION_SALES_PRICE	NULL
COLLECTION_INVENTORY_UNITS	NULL
COLLECTION_SALES_DOLLARS	NULL
COLLECTION_COST_OF_SALES	NULL

RDM Budget Data

The RDM_BUDGETS table maps to the P4P_BUDGET table.

Table 14–7 RDM Budget Data Mapping

RDM_BUDGETS	P4P_BUDGET
PI_ID	PI_ID
LOCATION_ID	LOCATION_ID
PERIOD_ID	PERIOD_ID
MARKDOWN_BUDGET	BUDGET
PLANNED_GM_AMT	PLANNED_GM_DOLLARS
PLANNED_GM_PERC	PLANNED_GM_PERC
FISCAL_MO	FISCAL_MONTH
FISCAL_YR	FISCAL_YR
BOH_UNITS	NULL (could be in RDM_BUDGETS_CDA_VW)
BOH_AMT	NULL (could be in RDM_BUDGETS_CDA_VW)
SALES_UNITS	NULL (could be in RDM_BUDGETS_CDA_VW)
SALES_AMT	NULL (could be in RDM_BUDGETS_CDA_VW)
TICKET_PRICE	NULL (could be in RDM_BUDGETS_CDA_VW)
OUTDATE	NULL (could be in RDM_BUDGETS_CDA_VW)

RDM Budget CDA Data Views

The RDM_BUDGETS_CDA_VW view maps to the PL_DD_ATTRIBUTES column where tablename='P4P_BUDGET' (bu=P4P_BUDGET).

Table 14–8 RDM CDA Budget Data View Mapping

RDM_BUDGETS_CDA_VW	PL_DD_ATTRIBUTES
PI_ID	bu.PI_ID PI_ID

Table 14–8 (Cont.) RDM CDA Budget Data View Mapping

RDM_BUDGETS_CDA_VW	PL_DD_ATTRIBUTES
LOCATION_ID	bu.LOCATION_ID LOCATION_ID
FISCAL_MONTH	bu.FISCAL_MONTH FISCAL_MONTH
FISCAL_YR	bu.FISCAL_YR FISCAL_YR
ATTRIBUTE1_NU_ NAME(1-10)	(select attributename from PL_DD_ATTRIBUTES where tablename='P4P_BUDGET' and columnname='ATTRIBUTE1') ATTRIBUTE1_NU_ NAME
ATTRIBUTE1 (1-10)	bu.ATTRIBUTE1 ATTRIBUTE1
ATTRIBUTE1_NU_ ISDISABLED (1-10)	(select isdisabled from PL_DD_ATTRIBUTES where tablename='P4P_BUDGET' and columnname='ATTRIBUTE1') ATTRIBUTE1_NU_ ISDISABLED

RDM Time-Period Data

The RDM_PERIODS_TBL table uses child aliases from the PERIODS_TBL table.

Table 14–9 RDM Time-Period Data Mapping

RDM_PERIODS_TBL	PERIODS_TBL
PERIOD_ID	PERIOD_ID
BEGIN_CALENDAR_DT	BEGIN_CALENDAR_DT
END_CALENDAR_DT	END_CALENDAR_DT
PERIOD_TYPE	PERIOD_TYPE
PERIOD_DESC	PERIOD_DESC
FISCAL_YR	FISCAL_YR
FISCAL_MO	FISCAL_MO
FISCAL_WK	FISCAL_WK
FISCAL_QUARTER	FISCAL_QUARTER
FISCAL_HALF	FISCAL_HALF
CALENDAR_YR	CALENDAR_YR
CALENDAR_MO	CALENDAR_MO
CALENDAR_WK	CALENDAR_WK
CALENDAR_QUARTER	CALENDAR_QUARTER
CALENDAR_HALF	CALENDAR_HALF
WK_PERIOD_ID	case when child.period_type in ('FD','FW') then (select period_id from periods_tbl parent where parent.begin_ calendar_dt <= child.begin_calendar_dt and parent.end_calendar_dt >= child.end_calendar_dt and parent.period_type = 'FW') end

Table 14–9 (Cont.) RDM Time-Period Data Mapping

RDM_PERIODS_TBL	PERIODS_TBL
MO_PERIOD_ID	case when child.period_type in ('FD','FW','FM') then (select period_id from periods_tbl parent where parent.begin_calendar_dt <= child.begin_calendar_dt and parent.end_calendar_dt >= child.end_calendar_dt and parent.period_type = 'FM') end
QUARTER_PERIOD_ID	case when child.period_type in ('FD','FW','FM','FQ') then (select period_id from periods_tbl parent where parent.begin_calendar_dt <= child.begin_calendar_dt and parent.end_calendar_dt >= child.end_calendar_dt and parent.period_type = 'FQ') end
HALF_PERIOD_ID	case when child.period_type in ('FD','FW','FM','FQ','FH') then (select period_id from periods_tbl parent where parent.begin_calendar_dt <= child.begin_calendar_dt and parent.end_calendar_dt >= child.end_calendar_dt and parent.period_type = 'FH') end
YR_PERIOD_ID	case when child.period_type in ('FD','FW','FM','FQ','FH','FY') then (select period_id from periods_tbl parent where parent.begin_calendar_dt <= child.begin_calendar_dt and parent.end_calendar_dt >= child.end_calendar_dt and parent.period_type = 'FY') end
HOLIDAY_FLAG	HOLIDAY_FLAG
HOLIDAY_DESC	HOLIDAY_DESC
SEASON	SEASON
SEASON_DESC	SEASON_DESC
SEASON_SEQ	NA. Unique to RDM.

RDM Merchandise Data

The RDM_MERCHANDISE_TBL table has synonym data pointing to Markdown Optimization.

Table 14–10 RDM Merchandise Data Mapping

RDM_MERCHANDISE_TBL	MERCHANDISE_TBL
PI_ID	PI_ID
MERCHANDISE_ID	MERCHANDISE_ID
PARENT_PI_ID	PARENT_PI_ID
PARENT_MERCHANDISE_ID	PARENT_MERCHANDISE_ID
CLIENT_LOAD_ID	CLIENT_LOAD_ID
HIERARCHY1_ID (1-15)	HIERARCHY1_ID (1-15)

Table 14–10 (Cont.) RDM Merchandise Data Mapping

RDM_MERCHANDISE_TBL	MERCHANDISE_TBL
HIERARCHY1_DESC (1-15)	HIERARCHY1_DESC (1-15)
HIERARCHY1_PID (1-15)	HIERARCHY1_PID (1-15)
HIERARCHY1_MID (1-15)	HIERARCHY1_MID (1-15)
MERCHANDISE_DESC	MERCHANDISE_DESC
BRAND	BRAND
BRAND_DESC	BRAND_DESC
ITEM_SIZE	ITEM_SIZE
REPORT_CLIENT_ID	REPORT_CLIENT_ID
START_DT	START_DT
END_DT	END_DT
FIRST_EFF_DT	FIRST_EFF_DT
LAST_EFF_DT	LAST_EFF_DT
LEVEL_SQC	LEVEL_SQC
LEVEL_DESC	LEVEL_DESC

RDM Merchandise CDA View Data

The RDM_MERCH_CDA_VW view maps to the PL_DD_ATTRIBUTES column where tablename='MERCH_ATTR_TBL' (mat=MERCH_ATTR_TBL).

Table 14–11 RDM Merchandise CDA View Mapping

RDM_MERCH_CDA_VW	PL_DD_ATTRIBUTES
PI_ID	mh.pi_id pi_id
ATTRIBUTE1_CH_NAME (1-8)	(select attributename from PL_DD_ATTRIBUTES where tablename='MERCH_ATTR_TBL' and columnname='ATTRIBUTE1') ATTRIBUTE1_CH_NAME
ATTRIBUTE1 (1-8)	mat.ATTRIBUTE1 ATTRIBUTE1
ATTRIBUTE1_CH_ISDISABLED (1-8)	(select isdisabled from PL_DD_ATTRIBUTES where tablename='MERCH_ATTR_TBL' and columnname='ATTRIBUTE1') ATTRIBUTE1_CH_ISDISABLED
ATTRIBUTE1_DT_NAME (1-8)	(select attributename from PL_DD_ATTRIBUTES where tablename='MERCH_ATTR_TBL' and columnname='ATTRIBUTE1_DATE') ATTRIBUTE1_DT_NAME
ATTRIBUTE1_DATE (1-8)	mat.ATTRIBUTE1_DATE ATTRIBUTE1_DATE
ATTRIBUTE1_DT_ISDISABLED (1-8)	(select isdisabled from PL_DD_ATTRIBUTES where tablename='MERCH_ATTR_TBL' and columnname='ATTRIBUTE1_DATE') ATTRIBUTE1_DT_ISDISABLED
ATTRIBUTE1_NU_NAME (1-8)	(select attributename from PL_DD_ATTRIBUTES where tablename='MERCH_ATTR_TBL' and columnname='ATTRIBUTE1_NUMBER') ATTRIBUTE1_NU_NAME

Table 14–11 (Cont.) RDM Merchandise CDA View Mapping

RDM_MERCH_CDA_VW	PL_DD_ATTRIBUTES
ATTRIBUTE1_NUMBER (1-8)	mat.ATTRIBUTE1_NUMBER ATTRIBUTE1_ NUMBER
ATTRIBUTE1_NU_ ISDISABLED (1-8)	(select isdisabled from PL_DD_ATTRIBUTES where tablename='MERCH_ATTR_TBL' and columnname='ATTRIBUTE1_NUMBER') ATTRIBUTE1_NU_ISDISABLED

RDM Location Data

The RDM_LOCATION_TBL table has synonyms pointing to Markdown Optimization geographic location data.

Table 14–12 RDM Location Data Mapping

RDM_LOCATION_TBL	LOCATION_TBL
LOCATION_ID	LOCATION_ID
HIERARCHY1_ID (1-12)	HIERARCHY1_ID (1-12)
HIERARCHY1_DESC (1-12)	HIERARCHY1_DESC (1-12)
HIERARCHY1_LID (1-12)	HIERARCHY1_LID (1-12)
LEVEL_DESC	LEVEL_DESC
LOCATION_DESC	LOCATION_DESC
STORE_CITY	STORE_CITY
STORE_STATE	STORE_STATE
STORE_ZIP	STORE_ZIP
VOLUME_GR	VOLUME_GR
STORE_CLASS	STORE_CLASS
MARKET_DESC	MARKET_DESC
NSLS_SQFT	NSLS_SQFT
GRS_ARE_SQFT	GRS_ARE_SQFT
START_DT	START_DT
END_DT	END_DT
FIRST_CREATE_DT	FIRST_CREATE_DT
LAST_MODIFIED_DT	LAST_MODIFIED_DT
STORE_DESC	STORE_DESC
GRSS_SQFT	GRSS_SQFT
CLIMATE	CLIMATE
STORE_FASHION_SEGMENT	STORE_FASHION_SEGMENT
STORE_AD_GROUP	STORE_AD_GROUP
STORE_SSC	STORE_SSC
SSC_IND	SSC_IND
STORE_CHST_1 (1-3)	STORE_CHST_1 (1-3)
FIRST_EFF_DT	FIRST_EFF_DT

Table 14–12 (Cont.) RDM Location Data Mapping

RDM_LOCATION_TBL	LOCATION_TBL
LAST_EFF_DT	LAST_EFF_DT
STORE_CLSS_IND	STORE_CLSS_IND
LOCATION_TYPE	LOCATION_TYPE
LEVEL_SQC	LEVEL_SQC
CLIENT_LOAD_ID	CLIENT_LOAD_ID
CLIENT_ID	CLIENT_ID

RDM Location CDA View Data

The RDM_LOCATION_CDA_VW maps to the PL_DD_ATTRIBUTES table where tablename='LOCATION_ATTR_TBL' (lat=LOCATION_ATTR_TBL).

Table 14–13 RDM Location CDA View Data Mapping

RDM_LOCATION_CDA_VW	PL_DD_ATTRIBUTES
LOCATION_ID	LOCATION_ID
ATTRIBUTE1_CH_NAME (1-8)	(select attributename from PL_DD_ATTRIBUTES where tablename='LOCATION_ATTR_TBL' and columnname='ATTRIBUTE1') ATTRIBUTE1_CH_NAME
ATTRIBUTE1 (1-8)	lat.ATTRIBUTE1 ATTRIBUTE1
ATTRIBUTE1_CH_ISDISABLED (1-8)	(select isdisabled from PL_DD_ATTRIBUTES where tablename='LOCATION_ATTR_TBL' and columnname='ATTRIBUTE1') ATTRIBUTE1_CH_ISDISABLED
ATTRIBUTE1_DT_NAME (1-8)	(select attributename from PL_DD_ATTRIBUTES where tablename='LOCATION_ATTR_TBL' and columnname='ATTRIBUTE1_DATE') ATTRIBUTE1_DT_NAME
ATTRIBUTE1_DATE (1-8)	lat.ATTRIBUTE1_DATE ATTRIBUTE1_DATE
ATTRIBUTE1_DT_ISDISABLED (1-8)	(select isdisabled from PL_DD_ATTRIBUTES where tablename='LOCATION_ATTR_TBL' and columnname='ATTRIBUTE1_DATE') ATTRIBUTE1_DT_ISDISABLED
ATTRIBUTE1_NU_NAME (1-8)	(select attributename from PL_DD_ATTRIBUTES where tablename='LOCATION_ATTR_TBL' and columnname='ATTRIBUTE1_NUMBER') ATTRIBUTE1_NU_NAME
ATTRIBUTE1_NUMBER	lat.ATTRIBUTE1_NUMBER ATTRIBUTE1_NUMBER
ATTRIBUTE1_NU_ISDISABLED (1-8)	(select isdisabled from PL_DD_ATTRIBUTES where tablename='LOCATION_ATTR_TBL' and columnname='ATTRIBUTE1_NUMBER') ATTRIBUTE1_NU_ISDISABLED

RDM Markdown History Data

The RDM_HIST_MARKDOWNS table is a view pointing to Markdown Optimization.

Table 14–14 RDM Markdown History Data Mapping

RDM_HIST_MARKDOWNS	HIST_MARKDOWNS_TBL
PI_ID	PI_ID
LOCATION_ID	LOCATION_ID
PERIOD_ID	PERIOD_ID
MARKDOWN_NUM	MARKDOWN_NUM
ACCEPTED_PRICE	ACCEPTED_PRICE
ACCEPTED_PCT_OFF	ACCEPTED_PCT_OFF
MD_TYPE	MD_TYPE
TEMP_ACCOUNTING_FG	TEMP_ACCOUNTING_FG
TAKEN_MARKDOWN_AMT	TAKEN_MARKDOWN_AMT
TAKEN_MARKDOWN	case when (accepted_price is not null) then 1 else 0 end

RDM System Tables

The following tables are used to report the status of the load/refresh tasks.

Table 14–15 RDM System Tables

RDM Table	RDM Table Description
RDM_SYSTEM_STATUS_TBL	Maintains a record for each of the latest weekly and incremental refreshes.
RDM_LOAD_STATUS_TBL	Maintains a detailed record for the tasks and subtasks that are part of the weekly and incremental refreshes.
RDM_TASK_LOOKUP_TBL	Maintains the master list of tasks and subtasks.
RDM_SYSTEM_DB	Maintains the information needed for the load scripts to recover after a failure.
RDM_MVIEWS	Maintains the information regarding the rollup tables that have been created. This information is used for the refreshes.

15

Metadata Metrics

This chapter lists the mapping between RDM data and OBIEE data.

It contains the following sections:

Overview

This chapter shows how RDM data maps to OBI EE data. The data is grouped in the following table according to the Logical Tables Item, Location, Merchandise, Time, Actuals Facts, Analysis Facts, Budget Facts, Forecast Facts, Historical Markdown Facts, and Item Data Facts. Within each group of Logical Tables, the Logical Columns are arranged alphabetically.

Metadata Metrics

The metadata includes the following metrics:

Table 15–1 Metadata Metrics

Name	Expression
Item	
Clearance Dt	RDM_ITEMS_TBL.CLEARANCE_DT
Clearance Ind Dt	RDM_ITEMS_TBL.CLEARANCE_IND_DT
Current Retail Begin Dt	RDM_ITEMS_TBL.CURRENT_RETAIL_BEGIN_DT
First Receipt Dt	RDM_ITEMS_TBL.FIRST_RECEIPT_DT
Item ID	RDM_ITEMS_TBL.ITEM_ID
Item Status	RDM_ITEMS_TBL.ITEM_STATUS
Item Vendor	RDM_ITEMS_TBL.VENDOR_ID
Merchandise ID	RDM_ITEMS_TBL.MERCHANDISE_ID
Season	RDM_ITEMS_TBL.SEASON_CODE
Location	
Location Hierarchy1 Desc	RDM_USER_LOCATIONS_VW.HIERARCHY1_DESC
Location Hierarchy1 ID	RDM_USER_LOCATIONS_VW.HIERARCHY1_LID
Location Hierarchy2 Desc	RDM_USER_LOCATIONS_VW.HIERARCHY2_DESC

Table 15–1 (Cont.) Metadata Metrics

Name	Expression
Location Hierarchy2 ID	RDM_USER_LOCATIONS_VW.HIERARCHY2_LID
Location Hierarchy3 Desc	RDM_USER_LOCATIONS_VW.HIERARCHY3_DESC
Location Hierarchy3 ID	RDM_USER_LOCATIONS_VW.HIERARCHY3_LID
Location Hierarchy4 Desc	RDM_USER_LOCATIONS_VW.HIERARCHY4_DESC
Location Hierarchy4 ID	RDM_USER_LOCATIONS_VW.HIERARCHY4_LID
Location Hierarchy5 Desc	RDM_USER_LOCATIONS_VW.HIERARCHY5_DESC
Location Hierarchy5 ID	RDM_USER_LOCATIONS_VW.HIERARCHY5_LID
Location Hierarchy6 Desc	RDM_USER_LOCATIONS_VW.HIERARCHY6_DESC
Location Hierarchy6 ID	RDM_USER_LOCATIONS_VW.HIERARCHY6_LID
Location Hierarchy7 Desc	RDM_USER_LOCATIONS_VW.HIERARCHY7_DESC
Location Hierarchy7 ID	RDM_USER_LOCATIONS_VW.HIERARCHY7_LID
Location Hierarchy8 Desc	RDM_USER_LOCATIONS_VW.HIERARCHY8_DESC
Location Hierarchy8 ID	RDM_USER_LOCATIONS_VW.HIERARCHY8_LID
Location Hierarchy9 Desc	RDM_USER_LOCATIONS_VW.HIERARCHY9_DESC
Location Hierarchy9 ID	RDM_USER_LOCATIONS_VW.HIERARCHY9_LID
Location Hierarchy10 Desc	RDM_USER_LOCATIONS_VW.HIERARCHY10_DESC
Location Hierarchy10 ID	RDM_USER_LOCATIONS_VW.HIERARCHY10_LID
Location Hierarchy11 Desc	RDM_USER_LOCATIONS_VW.HIERARCHY11_DESC
Location Hierarchy11 ID	RDM_USER_LOCATIONS_VW.HIERARCHY11_LID
Location Hierarchy12 Desc	RDM_USER_LOCATIONS_VW.HIERARCHY12_DESC
Location Hierarchy12 ID	RDM_USER_LOCATIONS_VW.HIERARCHY12_LID
Merchandise	
Product Hierarchy1 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY1_DESC
Product Hierarchy1 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY1_PID
Product Hierarchy2 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY2_DESC
Product Hierarchy2 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY2_PID
Product Hierarchy3 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY3_DESC
Product Hierarchy3 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY3_PID
Product Hierarchy4 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY4_DESC
Product Hierarchy4 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY4_PID
Product Hierarchy5 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY5_DESC
Product Hierarchy5 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY5_PID
Product Hierarchy6 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY6_DESC
Product Hierarchy6 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY6_PID
Product Hierarchy7 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY7_DESC
Product Hierarchy7 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY7_PID
Product Hierarchy8 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY8_DESC

Table 15–1 (Cont.) Metadata Metrics

Name	Expression
Product Hierarchy8 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY8_PID
Product Hierarchy9 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY9_DESC
Product Hierarchy9 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY9_PID
Product Hierarchy10 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY10_DESC
Product Hierarchy10 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY10_PID
Product Hierarchy11 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY11_DESC
Product Hierarchy11 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY11_PID
Product Hierarchy12 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY12_DESC
Product Hierarchy12 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY12_PID
Product Hierarchy13 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY13_DESC
Product Hierarchy13 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY13_PID
Product Hierarchy14 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY14_DESC
Product Hierarchy14 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY14_PID
Product Hierarchy15 Desc	RDM_USER_MERCHANDISE_VW.HIERARCHY15_DESC
Product Hierarchy15 ID	RDM_USER_MERCHANDISE_VW.HIERARCHY15_PID
Merchandise Desc	RDM_USER_MERCHANDISE_VW.MERCHANDISE_DESC
Merchandise ID	RDM_USER_MERCHANDISE_VW.MERCHANDISE_ID
Time	
Yr Period	RDM_PERIODS_TBL.YR_PERIOD_ID
Fiscal Yr	RDM_PERIODS_TBL.FISCAL_YR
Half Period	RDM_PERIODS_TBL.HALF_PERIOD_ID
Fiscal Half	RDM_PERIODS_TBL.FISCAL_HALF
Quarter Period	RDM_PERIODS_TBL.QUARTER_PERIOD_ID
Fiscal Quarter	RDM_PERIODS_TBL.FISCAL_QUARTER
Month Period	RDM_PERIODS_TBL.MO_PERIOD_ID
Fiscal Mo	RDM_PERIODS_TBL.FISCAL_MO
Wk Period	RDM_PERIODS_TBL.WK_PERIOD_ID
Fiscal Wk	RDM_PERIODS_TBL.FISCAL_WK
Actuals	
Act AUR	Sum(RDM_ACTIVITIES.NET_SALES_AMT) / Sum(RDM_ACTIVITIES.NET_SALES_UNITS)
Act Avg Full Price	Sum([RDM_ACTIVITIES.FULL_PRICE] *[RDM_ACTIVITIES.BOH_UNITS]) / Sum([RDM_ACTIVITIES.BOH_UNITS])
Act Avg Sales Price	Sum([RDM_ACTIVITIES.SALES_PRICE] * [RDM_ACTIVITIES.BOH_UNITS]) / Sum ([RDM_ACTIVITIES.BOH_UNITS])
Act Avg Selling Price	Sum([RDM_ACTIVITIES.NET_SALES_AMT]) / Sum([RDM_ACTIVITIES.NET_SALES_UNITS])
Act Avg Ticket Price	Sum([RDM_ACTIVITIES.TICKET_PRICE] * [RDM_ACTIVITIES.BOH_UNITS]) / Sum([RDM_ACTIVITIES.BOH_UNITS])

Table 15–1 (Cont.) Metadata Metrics

Name	Expression
Act BOH Units	Sum([RDM_ACTIVITIES.INVENTORY_UNITS] + [RDM_ACTIVITIES.NET_SALES_UNITS])
Act Delivered Units	Sum([RDM_ACTIVITIES.DELIVERED_UNITS])
Act GM Amt	Sum([RDM_ACTIVITIES.NET_SALES_AMT] - [RDM_ACTIVITIES.COST_OF_SALES])
Act GM%	[Act GM Amt] / [Act Sales Amt Net]
Act Inventory Units	Sum([RDM_ACTIVITIES.INVENTORY_UNITS])
Act Markdown Amt	Sum([RDM_HIST_MARKDOWNS.TAKEN_MARKDOWN_AMT])
Act OH Rtl Amt	[RDM_ACTIVITIES.NET_SALES_AMT] * [RDM_ACTIVITIES.NET_SALES_UNITS]
Act On Order Amt	Sum([RDM_ACTIVITIES.ITEM_COST] * [RDM_ACTIVITIES.ORDERED_UNITS])
Act On Order Units	Sum([RDM_ACTIVITIES.ORDERED_UNITS])
Act Pct Off TW	1 - ((Sum([RDM_ACTIVITIES.CURRENT_RETAIL] * [RDM_ACTIVITIES.BOH_UNITS] * [RDM_ACTIVITIES.FULL_PRICE]) / Sum([RDM_ACTIVITIES.FULL_PRICE] * [RDM_ACTIVITIES.BOH_UNITS] * [RDM_ACTIVITIES.FULL_PRICE]))
Act Returned Units	Sum([RDM_ACTIVITIES.RETURNED_UNITS])
Act Sales Amt Clearance	Sum([RDM_ACTIVITIES.CLR_SALES_AMT])
Act Sales Amt Full Price	Sum([RDM_ACTIVITIES.FULL_SALES_AMT])
Act Sales Amt Net	Sum([RDM_ACTIVITIES.NET_SALES_AMT])
Act Sales Amt POS	Sum([RDM_ACTIVITIES.POS_SALES_AMT])
Act Sales Cost Amt	Sum([RDM_ACTIVITIES.COST_OF_SALES])
Act Sales Units Clearance	Sum([RDM_ACTIVITIES.CLR_SALES_UNITS])
Act Sales Units Full Price	Sum([RDM_ACTIVITIES.FULL_SALES_UNITS])
Act Sales Units Net	Sum([RDM_ACTIVITIES.NET_SALES_UNITS])
Act Sales Units POS	Sum([RDM_ACTIVITIES.POS_SALES_UNITS])
Act Salvaged Units	Sum([RDM_ACTIVITIES.SALVAGED_UNITS])
Act Stock Sales Ratio	Sum([RDM_ACTIVITIES.BOH_UNITS]) / Sum([RDM_ACTIVITIES.NET_SALES_UNITS])
Analysis Metrics	
GM Amt vs Planned GM Amt	[Act GM Amt] - [Planned GM Amt]
GM% vs Planned GM%	[Act GM%] - [Planned GM Pct]
Sales Amt vs Planned Sales Amt	[Act Sales Amt Net] - [Planned Sales Amt]
Budget	
Planned AUR	Sum([RDM_BUDGETS.TICKET_PRICE] * [RDM_BUDGETS.SALES_UNITS]) / Sum([RDM_BUDGETS.SALES_UNITS])
Planned GM Amt	[RDM_BUDGETS.SALES_AMT] - ([RDM_BUDGETS.SALES_UNITS] * [RDM_ITEMS_TBL.UNIT_COST])
Planned GM Pct	[Planned GM Amt] / [Planned Sales Amt]

Table 15–1 (Cont.) Metadata Metrics

Name	Expression
Planned Markdown Amt	Sum([RDM_BUDGETS.MARKDOWN_BUDGET])
Planned Sales Amt	Sum([RDM_BUDGETS.SALES_AMT])
Planned Sales Cost	[Planned Sales Units] * [Avg Unit Cost]
Planned Sales Units	Sum([RDM_BUDGETS.SALES_UNITS])
Forecast	
Fcst Cost Sls	Sum([RDM_FORECAST_ACTIVITIES.COST_OF_SALES])
Fcst Grp Inv Units BOW	Sum([RDM_FORECAST_ACTIVITIES.COLLECTION_INVENTORY_UNITS] + [RDM_FORECAST_ACTIVITIES.COLLECTION_SALES_UNITS])
Fcst Grp Inv Units EOW	Sum([RDM_FORECAST_ACTIVITIES.COLLECTION_INVENTORY_UNITS])
Fcst Grp Sales Price	Sum([RDM_FORECAST_ACTIVITIES.COLLECTION_INVENTORY_UNITS] * [RDM_FORECAST_ACTIVITIES.COLLECTION_SALES_PRICE]) / Sum([RDM_FORECAST_ACTIVITIES.COLLECTION_INVENTORY_UNITS])
Fcst Grp Sales Units	Sum([RDM_FORECAST_ACTIVITIES.COLLECTION_SALES_UNITS])
Fcst Grp Stock Sales Ratio	Sum([RDM_FORECAST_ACTIVITIES.COLLECTION_INVENTORY_UNITS] + [RDM_FORECAST_ACTIVITIES.COLLECTION_SALES_UNITS]) / Sum([RDM_FORECAST_ACTIVITIES.COLLECTION_SALES_UNITS])
Fcst Grp Ticket Price	Sum([RDM_FORECAST_ACTIVITIES.COLLECTION_INVENTORY_UNITS] * [RDM_FORECAST_ACTIVITIES.COLLECTION_TICKET_PRICE]) / Sum([RDM_FORECAST_ACTIVITIES.COLLECTION_INVENTORY_UNITS])
Fcst Grp WOS	Sum([RDM_FORECAST_ACTIVITIES.COLLECTION_INVENTORY_UNITS] * [RDM_FORECAST_ACTIVITIES.COLLECTION_SALES_UNITS]) / Sum([RDM_FORECAST_ACTIVITIES.COLLECTION_SALES_UNITS])
Fcst Inv Units BOW	Sum([RDM_FORECAST_ACTIVITIES.INVENTORY_UNITS] + [RDM_FORECAST_ACTIVITIES.SALES_UNITS])
Fcst Inv Units EOW	Sum([RDM_FORECAST_ACTIVITIES.INVENTORY_UNITS])
Fcst Sales Amt	Sum([RDM_FORECAST_ACTIVITIES.SALES_UNITS] * {RDM_FORECAST_ACTIVITIES.SALES_PRICE])
Fcst Sales Price	Sum([RDM_FORECAST_ACTIVITIES.INVENTORY_UNITS] * {RDM_FORECAST_ACTIVITIES.SALES_PRICE]) / Sum([RDM_FORECAST_ACTIVITIES.INVENTORY_UNITS])
Fcst Sales Units	Sum([RDM_FORECAST_ACTIVITIES.SALES_UNITS])
Fcst Stock Sales Ratio	Sum([RDM_FORECADT_ACTIVITIES.INVENTORY_UNITS] + [RDM_FORECAST_ACTIVITIES.SALES_UNITS]) / Sum([RDM_FORECAST_ACTIVITIES.SALES_UNITS])
Fcst Ticket Price	Sum([RDM_FORECAST_ACTIVITIES.TICKET_PRICE] * [RDM_FORECAST_ACTIVITIES.INVENTORY_UNITS]) / Sum([RDM_FORECAST_ACTIVITIES.INVENTORY_UNITS])
Fcst WOS	Sum([RDM_FORECAST_ACTIVITIES.INVENTORY_UNITS] * [RDM_FORECAST_ACTIVITIES.SALES_UNITS]) / Sum([RDM_FORECAST_ACTIVITIES.SALES_UNITS])
Hist Markdowns	
Hist Accepted Pct Off	Avg([RDM_HIST_MARKDOWNS.ACCEPTED_PCT_OFF])
Hist Taken MD Amt	Sum([RDM_HIST_MARKDOWNS.TAKEN_MARKDOWN])
MarkDown Count	Count([RDM_HIST_MARKDOWNS.MARKDOWN_NUM])
Item Data	

Table 15–1 (Cont.) Metadata Metrics

Name	Expression
Accepted MDs	Sum([RDM_ITEM_DATA.TAKEN_MARKDOWN] * [RDM_ITEM_DATA.ACCEPTED_MARKDOWN])
Added MD Amt	Sum(([RDM_ITEM_DATA.NON_REC_MARKDOWN] * [RDM_ITEM_DATA.TAKEN_MARKDOWN]) * [RDM_ITEM_DATA.TAKEN_MARKDOWN]) * [RDM_ITEM_DATA.PROJ_UNITS_OH_NW] * ([RDM_ITEM_DATA.CURRENT_RTL_PRICE] - [RDM_ITEM_DATA>ACCEPTED_PRICE]))
Added MD Cost Amt	Sum(([RDM_ITEM_DATA.NON_REC_MARKDOWN] * [RDM_ITEM_DATA.TAKEN_MARKDOWN]) * ([RDM_ITEM_DATA.PROJ_UNITS_OH_NW] * [RDM_ITEM_DATA.UNIT_COST]))
Added MDs	Sum([RDM_ITEM_DATA.NON_REC_MARKDOWN] * [RDM_ITEM_DATA.TAKEN_MARKDOWN])
Added Unit Inv	Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS] * ([RDM_ITEM_DATA.NON_REC_MARKDOWN] * [RDM_ITEM_DATA.TAKEN_MARKDOWN]))
AUC	Sum([RDM_ITEM_DATA.UNIT_COST] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
AUR LLW	Sum([RDM_ITEM_DATA.SALES_AMT_WM1] / Sum([RDM_ITEM_DATA.SALES_UNITS_WM1])
AUR LTD	Sum([RDM_ITEM_DATA.SALES_AMT_LTD]) / Sum([RDM_ITEM_DATA.SALES_UNITS_LTD])
AUR LW	Sum([RDM_ITEM_DATA.SALES_AMT]) / Sum([RDM_ITEM_DATA.SALES_UNITS])
AUR NW	Sum([RDM_ITEM_DATA.FA_SALES_DOLLARS_NW]) / Sum([RDM_ITEM_DATA.FA_SALES_UNITS_NW])
AUR TW	Sum([RDM_ITEM_DATA.FA_SALES_DOLLARS_TW]) / Sum([RDM_ITEM_DATA.FA_SALES_UNITS_TW])
Avg Accepted Price	Sum([RDM_ITEM_DATA.ACCEPTED_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.INV_UNITS_OH])
Avg Cur Rtl Pct Off Orig Rtl	Avg([RDM_ITEM_DATA.CURRTL_PERC_OFF_ORRTL])
Avg FTB Ratio	Avg([RDM_ITEM_DATA.FTB_RATIO])
Avg Markup Pct	Avg([RDM_ITEM_DATA.MARKUP_PERC])
Avg Next Rtl Price	Avg([RDM_ITEM_DATA.NEXT_RTL_PRICE])
Avg Owned Rtl Price	Avg([RDM_ITEM_DATA.OWNED_RTL_PRICE])
Avg Price	Sum([RDM_ITEM_DATA.CURRENT_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Avg Price LTD	Sum([RDM_ITEM_DATA.AVG_PRICE_LTD] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Avg Rec Pct Off Cur Rtl	Avg([RDM_ITEM_DATA.REC_PERC_OFF_CURTL])
Avg Rec Pct Off Orig Rtl	Avg([RDM_ITEM_DATA.REC_PERC_OFF_ORRTL])
Avg Rec Rtl Price	Sum([RDM_ITEM_DATA.REC_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Avg Taken Pct Off Cur Rtl	Avg([RDM_ITEM_DATA.TAKEN_PERC_OFF_CURTL])

Table 15–1 (Cont.) Metadata Metrics

Name	Expression
Avg Taken Pct Off Orig Rtl	Avg([RDM_ITEM_DATA.TAKEN_PERC_OFF_ORRTL])
Avg Target Inv Units	Avg([RDM_ITEM_DATA.TARGET_INVENTORY_UNITS])
Avg Unit Cost	Avg([RDM_ITEM_DATA.UNIT_COST])
Chain Avg Price	Sum([RDM_ITEM_DATA.AVG_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Chain Max Price	Max([RDM_ITEM_DATA.CHAIN_MAX_PRICE])
Chain Min Price	Min([RDM_ITEM_DATA.CHAIN_MIN_PRICE])
Cost Sls (TW-EOL)	Sum(([RDM_ITEM_DATA.PROJ_SALES_UNITS_EOL] - [RDM_ITEM_DATA.SALES_UNITS_LTD]) * [RDM_ITEM_DATA.UNIT_COST]))
Cost Sls LTD	Sum([RDM_ITEM_DATA.SALES_UNITS_LTD] * [RDM_ITEM_DATA.UNIT_COST])
Cost Sls LW	Sum([RDM_ITEM_DATA.SALES_UNITS] * [RDM_ITEM_DATA.UNIT_COST])
Cost Sls NW	Sum([RDM_ITEM_DATA.FA_SALES_UNITS_NW] * [RDM_ITEM_DATA.UNIT_COST])
Cost Sls TW	Sum([RDM_ITEM_DATA.FA_SALES_UNITS_TW] * [RDM_ITEM_DATA.UNIT_COST])
Cur Rtl Price	Sum([RDM_ITEM_DATA.CURRENT_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Current Cost Amt	Sum([RDM_ITEM_DATA.CURRENT_COST_AMT])
Date Sent	Max([RDM_ITEM_DATA.SENT_DATE])
DC OH Units	Sum([RDM_ITEM_DATA.DC_OH_UNITS])
DC OO Units	Sum([RDM_ITEM_DATA.DC_OO_UNITS])
Delayed MDs	Sum([RDM_ITEM_DATA.DELAYED_MARKDOWN])
Exit Date	Min([RDM_ITEM_DATA.PROJ_OUT_OF_STOCK])
Exit Date Last Mod Date	Max([RDM_ITEM_DATA.ORIGINAL_EXIT_DATE_MOD_DT])
First MDs	Sum([RDM_ITEM_DATA.IS_FIRST_MD])
First Rcpt Dt	Max([RDM_ITEM_DATAFIRST_RECEIPT_DATE])
First Sale Date	Min([RDM_ITEM_DATA.FIRST_SALE_DATE])
Full Price	Avg([RDM_ITEMS_TBL.FULL_PRICE])
GM Amt LTD	Sum([RDM_ITEM_DATA.SALES_AMT_LTD] - ([RDM_ITEM_DATA.UNIT_COST] * [RDM_ITEM_DATA.SALES_UNITS_LTD]))
GM Pct LTD	[GM Amt LTD] / [Sales Amt LTD]
Gross Profit Amt	Sum([RDM_ACTIVITIES.NET_SALES_AMT] - [RDM_ACTIVITIES.COST_OF_SALES])
Gross Profit Amt LTD	Sum([RDM_ITEM_DATA.GROSS_PROFIT_AMT_LTD])
Group MDs	Sum([RDM_ITEM_DATA.REC_AS_COLLECTION])
Inv Units EOW LLLLW	Sum([RDM_ITEM_DATA.INV_UNITS_WM3])
Inv Units EOW LLLW	Sum([RDM_ITEM_DATA.INV_UNITS_WM2])
Inv Units EOW LLW	Sum([RDM_ITEM_DATA.INV_UNITS_WM1])

Table 15–1 (Cont.) Metadata Metrics

Name	Expression
Inv Units EOW LW	Sum([RDM_ITEM_DATA.INV_UNITS_OH])
Item MDs	Sum([RDM_ITEM_DATA.REC_AS_ITEM])
Last MD Date	Max([RDM_ITEM_DATA.LAST_MD_EFFECTIVE_DT])
Last Rcpt Dt	Max([RDM_ITEM_DATA.LAST_RECEIPT_DATE])
Lost Opportunity Cost	Sum([RDM_ITEM_DATA.OPPORTUNITY_COST] * (1 - [RDM_ITEM_DATA.TAKEN_MARKDOWN]))
Lost Opportunity Cost(Grp)	Sum(([RDM_ITEM_DATA.OPPORTUNITY_COST] * [RDM_ITEM_DATA.AS_COLLECTION]) * (1 - [RDM_ITEM_DATA.TAKEN_MARKDOWN]))
Max MD Number	Max([RDM_ITEM_DATA.MD_NUMBER])
MD Effective Date	Min([RDM_ITEM_DATA.EFFECTIVE_DATE])
Mod Not Acc MDs	Sum([RDM_ITEM_DATA.MODIFIED_MARKDOWN])
Modified Exit Date	Max([RDM_ITEM_DATA.MODIFIED_OUT_OF_STOCK_DATE])
Modified MDs	Sum([RDM_ITEM_DATA.MODIFIED_MARKDOWN] * [RDM_ITEM_DATA.TAKEN_MARKDOWN])
Modified Target ST Pct	Sum([RDM_ITEM_DATA.MODIFIED_TARGET_ST_PERC] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
MU Pct Init	Sum(([RDM_ITEM_DATA.ORIGINAL_RTL_PRICE] - [RDM_ITEM_DATA.UNIT_COST]) * [RDM_ITEM_DATA.COMMITTED_INV_UNITS])) / Sum([RDM_ITEM_DATA.ORIGINAL_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS])
MU Pct NW	Sum(([RDM_ITEM_DATA.FA_TICKET_PRICE_NW] - [RDM_ITEM_DATA.UNIT_COST]) * [RDM_ITEM_DATA.FA_INVENTORY_UNITS_NW]) / Sum([RDM_ITEM_DATA.FA_TICKET_PRICE_NW] * [RDM_ITEM_DATA.FA_INVENTORY_UNITS_NW])
MU Pct TW	Sum(([RDM_ITEM_DATA.CURRENT_RTL_PRICE] - [RDM_ITEM_DATA.UNIT_COST]) * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.CURRENT_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Next Rec Date	Min([RDM_ITEM_DATA.NEXT_MD_DATA])
Next Rec Pct Off	Max([RDM_ITEM_DATA.NEXT_REC_PERC_OFF_ORRTL])
Num Items	Count([RDM_ITEM_DATA.UNIT_COST])
Num Stores OH	Sum([RDM_ITEM_DATA.NO_STORE_WITH_OH])
OH Cost Amt	Sum([RDM_ITEM_DATA.INV_COST_AMT_OH])
OH Rtl Amt	Sum([RDM_ITEM_DATA.CURRENT_RTL_PRICE] / [RDM_ITEM_DATA.INV_UNITS_OH])
On Hand Inv	Sum([RDM_ITEM_DATA.INV_UNITS_OH])
On Order Inv	Sum([RDM_ITEM_DATA.INV_UNITS_OO])
OO Cost Amt	Sum([RDM_ITEM_DATA.INV_COST_AMT_OH])
OO Rtl Amt	Sum([RDM_ITEM_DATA.CURRENT_RTL_PRICE] * [RDM_ITEM_DATA.INV_UNITS_OO])
Opportunity Cost	Sum([RDM_ITEM_DATA.OPPORTUNITY_COST])
Opportunity Cost(Grp)	Sum([RDM_ITEM_DATA.OPPORTUNITY_COST] * [RDM_ITEM_DATA.REC_AS_COLLECTION])

Table 15–1 (Cont.) Metadata Metrics

Name	Expression
Orig Rtl Price	Sum([RDM_ITEM_DATA.ORIGINAL_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Original Exit Date	Max([RDM_ITEM_DATA.ORIGINAL_EXIT_DATE])
OTB MDs	Sum([RDM_ITEM_DATA.MD_TAKEN_THRO_OTB])
OTB Taken MDs	Sum([RDM_ITEM_DATA.MD_TAKEN_THRU_OTB] * [RDM_ITEM_DATA.TEMP_MD_FLAG])
Owned Rtl	Sum([RDM_ITEM_DATA.OWNED_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Pct Tlt Inv Allocated	Sum([RDM_ITEM_DATA.SALES_UNITS_LTD] + [RDM_ITEM_DATA.INV_UNITS_OH]) + [RDM_ITEM_DATA.INV_UNITS_OO]) / Sum([RDM_ITEM_DATA.SALES_UNITS_LTD] + [RDM_ITEM_DATA.INV_UNITS_OH] + [RDM_ITEM_DATA.INV_UNITS_OO] + [RDM_ITEM_DATA.DC_OH_UNITS] + [RDM_ITEM_DATA.DC_OO_UNITS])
Perm MDs	Sum(1 - [RDM_ITEM_DATA.TEMP_MD_FLAG])
Planned Start Date	Min([RDM_ITEM_DATA.PLANNED_START_SELL_DATE])
Price Ladder Description	Max([RDM_ITEM_DATA.PRICE_LADDER_DESC])
Price Ladder Sent	Min([RDM_ITEM_DATA.SENT_LADDER_ID])
Price Ladder Type	Max([RDM_ITEM_DATA.PRICE_LADDER_TYPE])
Price Sent	Sum([RDM_ITEM_DATA.SENT_MARKDOWN_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Proj Cost Sls EOL	Sum([RDM_ITEM_DATA.UNIT_COST] * [RDM_ITEM_DATA.PROJ_SALES_UNITS_EOL])
Proj Exit Date	Min([RDM_ITEM_DATA.PROJ_OUT_OF_STOCK])
Proj GM Amt EOL(Grp)	Sum([RDM_ITEM_DATA.PROJ_GM_AMT_EOL_GRP])
Proj GM Amt EOL(Item)	Sum([RDM_ITEM_DATA.PROJ_GM_AMT_EOL])
Proj GM Amt NW	Sum([RDM_ITEM_DATA.FA_SALES_DOLLARS_NW] - ([RDM_ITEM_DATA.FA_SALES_UNITS_NW] * [RDM_ITEM_DATA.UNIT_COST]))
Proj GM Amt TW	Sum([RDM_ITEM_DATA.FA_SALES_DOLLARS_TW] - ([RDM_ITEM_DATA.FA_SALES_UNITS_TW] * [RDM_ITEM_DATA.UNIT_COST]))
Proj GM Pct EOL	Sum([RDM_ITEM_DATA.PROJ_GM_PERC_EOL] * [RDM_ITEM_DATA.PROJ_SALES_AMT_EOL]) / Sum([RDM_ITEM_DATA.PROJ_SALES_AMT_EOL])
Proj GM Pct EOL(Grp)	Sum([RDM_ITEM_DATA.PROJ_GM_AMT_EOL_GRP] / [RDM_ITEM_DATA.PROJ_SALES_AMT_EOL]) / Sum([RDM_ITEM_DATA.PROJ_SALES_AMT_EOL])
Proj MU Pct EOL	Avg (([RDM_ITEM_DATA.REC_RTL_MIN] - [RDM_ITEM_DATA.UNIT_COST]) / [RDM_ITEM_DATA.REC_RTL_MIN])
Proj OH Cost Amt EOL	Sum([RDM_ITEM_DATA.ENDING_INV_UNITS] * [RDM_ITEM_DATA.UNIT_COST])
Proj OH EOL	Sum([RDM_ITEM_DATA.ENDING_INV_UNITS])
Proj OH NW	Sum([RDM_ITEM_DATA.PROJ_UNITS_OH_NW])
Proj OH Rtl Amt EOL	Sum([RDM_ITEM_DATA.ENDING_INV_UNITS] * [RDM_ITEM_DATA.PROJ_OH_RTL_EOL])

Table 15–1 (Cont.) Metadata Metrics

Name	Expression
Proj OH Rtl EOL	Sum([RDM_ITEM_DATA.PROJ_OH_RTL_EOL])
Proj Out Of Stock Dt	Min([RDM_ITEM_DATA.PROJ_OUT_OF_STOCK])
Proj Rtl Price EOL	Sum([RDM_ITEM_DATA.REC_RTL_MIN] * [RDM_ITEM_DATA.ENDING_INV_UNITS]) / Sum([RDM_ITEM_DATA.ENDING_INV_UNITS])
Proj Sales Amt EOL	Sum([RDM_ITEM_DATA.PROJ_SALES_AMT_EOL])
Proj Sales Units EOL	Sum([RDM_ITEM_DATA.PROJ_SALES_UNITS_EOL])
Proj ST Pct EOL	[RDM_ITEM_DATA.PROJ_SALES_UNITS_EOL] / ([RDM_ITEM_DATA.PROJ_SALES_UNITS_EOL] + [RDM_ITEM_DATA.ENDING_INV_UNITS])
Promo Desc	Min([RDM_ITEM_DATA.PROMO_DESC])
Promo End Dt	Max([RDM_ITEM_DATA.PROMO_END_DT])
Promo Flag	Max([RDM_ITEM_DATA.PROMO_FLAG])
Promo Pct Off	Max([RDM_ITEM_DATA.PROMO_PCT_OFF])
Promo Rtl	Min([RDM_ITEM_DATA.LOWEST_PROMO_PRICE])
Promo Start Dt	Min([RDM_ITEM_DATA.PROMO_START_DT])
Rec Md Amt	Sum(([RDM_ITEM_DATA.RECOMMENDED_MARKDOWN] * [RDM_ITEM_DATA.PROJ_UNITS_OH_NW]) * ([RDM_ITEM_DATA.CURRENT_RTL_PRICE] - [RDM_ITEM_DATA.REC_RTL_PRICE]))
Rec MD Amt Cost	Sum(([RDM_ITEM_DATA.RECOMMENDED_MARKDOWN] * [RDM_ITEM_DATA.PROJ_UNITS_OH_NW]) * [RDM_ITEM_DATA.UNIT_COST])
Rec MD Amt Cost Not Taken	Sum([RDM_ITEM_DATA.RECOMMENDED_MARKDOWN] * (1 - [RDM_ITEM_DATA.RECOMMENDED_MARKDOWN] * [RDM_ITEM_DATA.PROJ_UNITS_OH_NW] * [RDM_ITEM_DATA.UNIT_COST]))
Rec MD Amt Not Taken	Sum(([RDM_ITEM_DATA.RECOMMENDED_MARKDOWN] * (1 - [RDM_ITEM_DATA.RECOMMENDED_MARKDOWN] * [RDM_ITEM_DATA.PROJ_UNITS_OH_NW]) * ([RDM_ITEM_DATA.CURRENT_RTL_PRICE] - [RDM_ITEM_DATA.REC_RTL_PRICE]))
Rec MD Inv	Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS] * [RDM_ITEM_DATA.RECOMMENDED_MARKDOWN])
Rec Md Inv Cost	Sum([RDM_ITEM_DATA.REC_MD_INV_COST])
Rec MDs	Sum([RDM_ITEM_DATA.RECOMMENDED_MARKDOWN])
Rec MDs (Group)	Sum([RDM_ITEM_DATA.REC_AS_COLLECTION])
Rec MDs (Item)	Sum([RDM_ITEM_DATA.REC_AS_ITEM])
Rec MU Pct	Sum(([RDM_ITEM_DATA.REC_RTL_PRICE] - [RDM_ITEM_DATA.UNIT_COST]) * [RDM_ITEM_DATA.REC_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.REC_RTL_PRICE] * [RDM_ITEM_DATA.REC_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Rec Pct Off Curr	ZeroToNull(Sum(([RDM_ITEM_DATA.CURRENT_RTL_PRICE] - [RDM_ITEM_DATA.REC_RTL_PRICE]) / ([RDM_ITEM_DATA.CURRENT_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS])) / Sum([RDM_ITEM_DATA.CURRENT_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]))

Table 15–1 (Cont.) Metadata Metrics

Name	Expression
Rec Pct Off Orig	ZeroToSum(Sum(([RDM_ITEM_DATA.ORIGINAL_RTL_PRICE] - [RDM_ITEM_DATA.REC_RTL_PRICE]) / ([RDM_ITEM_DATA.ORIGINAL_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum(([RDM_ITEM_DATA.ORIGINAL_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]))
Rec Rtl (Grp)	Sum([RDM_ITEM_DATA.REC_COLLECTION_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Rec Rtl (Item)	ZeroToNull(Sum([RDM_ITEM_DATA.REC_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS]))
Rec Rtl Max	Max([RDM_ITEM_DATA.REC_RTL_PRICE])
Rec Rtl Min	Min([RDM_ITEM_DATA.REC_RTL_PRICE])
Rec Unit Inv	Sum([RDM_ITEM_DATA.PROJ_UNITS_OH_NW] * [RDM_ITEM_DATA.RECOMMENDED_MARKDOWN])
Rec Units Not Taken	Sum(([RDM_ITEM_DATA.PROJ_UNITS_OH_NW] * [RDM_ITEM_DATA.RECOMMENDED_MARKDOWN]) * (1 - [RDM_ITEM_DATA.RECOMMENDED_MARKDOWN]))
Sales Amt (TW-EOL)	[RDM_ITEM_DATA.PROJ_SALES_AMT_EOL] - [RDM_ITEM_DATA.SALES_AMT_LTD]
Sales Amt LLLLW	Sum([RDM_ITEM_DATA.SALES_AMT_WM3])
Sales Amt LLLW	Sum([RDM_ITEM_DATA.SALES_AMT_WM2])
Sales Amt LLW	Sum([RDM_ITEM_DATA.SALES_AMT_WM1])
Sales Amt LTD	Sum([RDM_ITEM_DATA.SALES_AMT_LTD])
Sales Amt LW	Sum([RDM_ITEM_DATA.SALES_AMT])
Sales Amt NW	Sum([RDM_ITEM_DATA.FA_SALES_DOLLARS_NW])
Sales Amt TW	Sum([RDM_ITEM_DATA.FA_SALES_DOLLARS_TW])
Sales Units (TW-EOL)	[RDM_ITEM_DATA.PROJ_SALES_UNITS_EOL] - [RDM_ITEM_DATA.SALES_UNIT_LTD]
Sales Units LLLLW	Sum([RDM_ITEM_DATA.SALES_UNITS_WM3])
Sales Units LLLW	Sum([RDM_ITEM_DATA.SALES_UNITS_WM2])
Sales Units LLW	Sum([RDM_ITEM_DATA.SALES_UNITS_WM1])
Sales Units LTD	Sum([RDM_ITEM_DATA.SALES_UNITS_LTD])
Sales Units LW	Sum([RDM_ITEM_DATA.SALES_UNITS])
Sales Units NW	Sum([RDM_ITEM_DATA.FA_SALES_UNITS_NW])
Sales Units TW	Sum([RDM_ITEM_DATA.FA_SALES_UNITS_TW])
Salvage Amt	Sum([RDM_ITEM_DATA.SALVAGE_VALUE_AMT])
Salvage Pct	Sum([RDM_ITEM_DATA.SALVAGE_VALUE_PERC] * [RDM_ITEM_DATA.ENDING_INV_UNITS]) / Sum([RDM_ITEM_DATA.ENDING_INV_UNITS])
Sendback Date	Max([RDM_ITEM_DATA.SENDBACK_DATE])
ST Pct (TW-EOL)	[RDM_ITEM_DATA.PROJ_SALES_UNITS_EOL] / [RDM_ITEM_DATA.SALES_UNITS_LTD]

Table 15–1 (Cont.) Metadata Metrics

Name	Expression
ST Pct LLLLW	Sum([RDM_ITEM_DATA.SELL_THROUGH_WM3] * ([RDM_ITEM_DATA.SALES_UNITS_WM3] + [RDM_ITEM_DATA.INV_UNITS_WM3])) / Sum([RDM_ITEM_DATA.SALES_UNITS_WM3] + [RDM_ITEM_DATA.INV_UNITS_WM3])
ST Pct LLLW	Sum([RDM_ITEM_DATA.SELL_THROUGH_WM2] * ([RDM_ITEM_DATA.SALES_UNITS_WM2] + [RDM_ITEM_DATA.INV_UNITS_WM2])) / Sum([RDM_ITEM_DATA.SALES_UNITS_WM2] + [RDM_ITEM_DATA.INV_UNITS_WM2])
ST Pct LTD	Sum([RDM_ITEM_DATA.SELL_THROUGH_LTD] * ([RDM_ITEM_DATA.SALES_UNITS_LTD] + [RDM_ITEM_DATA.INV_UNITS_OH])) / Sum([RDM_ITEM_DATA.SELL_THROUGH_LTD] + [RDM_ITEM_DATA.INV_UNITS_OH])
ST Pct LW	Sum([RDM_ITEM_DATA.SELL_THROUGH] * ([RDM_ITEM_DATA.SALES_UNITS] + [RDM_ITEM_DATA.INV_UNITS_OH])) / Sum([RDM_ITEM_DATA.SALES_UNITS] + [RDM_ITEM_DATA.INV_UNITS_OH])
Start Date	Min([RDM_ITEM_DATA.START_SELL_DATE])
Stock Sales Ratio TW	Sum([RDM_ITEM_DATA.INV_UNITS_OH]) / Sum([RDM_ITEM_DATA.FA_SALES_UNITS_TW])
Taken Curr Rtl Value	Sum([RDM_ITEM_DATA.TAKEN_MARKDOWN] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS] * [RDM_ITEM_DATA.CURRENT_RTL_PRICE])
Taken Deeper MDs	Sum([RDM_ITEM_DATA.TAKEN_DEEPER] * [RDM_ITEM_DATA.TAKEN_MARKDOWN])
Taken MD Amt (Perm)	Sum(([RDM_ITEM_DATA.TAKEN_MARKDOWN] * (1 - [RDM_ITEM_DATA.TEMP_MD_FLAG])) * [RDM_ITEM_DATA.PROJ_UNITS_OH_NW] * ([RDM_ITEM_DATA.CURRENT_RTL_PRICE] - [RDM_ITEM_DATA.REC_RTL_PRICE]))
Taken MD Amt (Temp)	Sum([RDM_ITEM_DATA.TAKEN_MARKDOWN_FLAG] * [RDM_ITEM_DATA.TEMP_MD_FLAG] * [RDM_ITEM_DATA.PROJ_UNITS_OH_NW] * ([RDM_ITEM_DATA.CURRENT_RTL_PRICE] - [RDM_ITEM_DATA.REC_RTL_PRICE]))
Taken MD Amt (Total)	Sum([RDM_ITEM_DATA.TAKEN_MARKDOWN_FLAG] * [RDM_ITEM_DATA.PROJ_UNITS_OH_NW] * ([RDM_ITEM_DATA.CURRENT_RTL_PRICE] - [RDM_ITEM_DATA.ACCEPTED_PRICE]))
Taken MD Amt Cost (Perm)	Sum(([RDM_ITEM_DATA.TAKEN_MARKDOWN] * (1 - [RDM_ITEM_DATA.TEMP_MD_FLAG]) * ([RDM_ITEM_DATA.PROJ_UNITS_OH_NW] * [RDM_ITEM_DATA.UNIT_COST]))
Taken MD Amt Cost (Temp)	Sum([RDM_ITEM_DATA.TAKEN_MARKDOWN] * [RDM_ITEM_DATA.TEMP_MD_FLAG] * [RDM_ITEM_DATA.PROJ_UNITS_OH_NW] * [RDM_ITEM_DATA.UNIT_COST])
Taken MD Amt Cost (Total)	[Taken MD Amt Cost (Perm)] + [Taken MD Amt Cost (Temp)]
Taken Md Inv Cost	Sum([RDM_ITEM_DATA.TAKEN_MD_INV_COST])
Taken MDs	Sum([RDM_ITEM_DATA.TAKEN_MARKDOWN])
Taken Mod MDs	Sum([RDM_ITEM_DATA.TAKEN_MARKDOWN] * [RDM_ITEM_DATA.MODIFIED_MARKDOWN])
Taken MU Pct	Sum(([RDM_ITEM_DATA.TAKEN_MD_PRICE] - [RDM_ITEM_DATA.UNIT_COST]) * [RDM_ITEM_DATA.COMMITTED_INV_UNITS] * [RDM_ITEM_DATA.TAKEN_MD_PRICE]) / Sum([RDM_ITEM_DATA.REC_RTL_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS] * [RDM_ITEM_DATA.TAKEN_MD_PRICE])

Table 15–1 (Cont.) Metadata Metrics

Name	Expression
Taken Pct Off Curr	Sum([RDM_ITEM_DATA.TAKEN_MARKDOWN] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) * [RDM_ITEM_DATA.CURRENT_RTL_PRICE] * [RDM_ITEM_DATA.TAKEN_PERC_OFF_CURTL]) / Sum([RDM_ITEM_DATA.TAKEN_MARKDOWN] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS] * [RDM_ITEM_DATA.CURRENT_RTL_PRICE])
Taken Pct Off Orig	Sum([RDM_ITEM_DATA.TAKEN_MARKDOWN] * [RDM_ITEM_DATA.TAKEN_PERC_OFF_ORRTL]) * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) * [RDM_ITEM_DATA.ORIGINAL_RTL_PRICE]) / Sum([RDM_ITEM_DATA.TAKEN_MARKDOWN] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS] * [RDM_ITEM_DATA.ORIGINAL_RTL_PRICE])
Taken Rec MDs	Sum([RDM_ITEM_DATA.TAKEN_MARKDOWN] * [RDM_ITEM_DATA.ACCEPTED_MARKDOWN])
Taken Rtl	Sum([RDM_ITEM_DATA.TAKEN_MD_PRICE] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Taken Shallower MDs	Sum([RDM_ITEM_DATA.TAKEN_SHALLOWER] * [RDM_ITEM_DATA.TAKEN_MARKDOWN])
Taken Unit Inv	Sum([RDM_ITEM_DATA.PROJ_UNITS_OH_NW] * [RDM_ITEM_DATA.TAKEN_MARKDOWN])
Target St Inv Units	Sum([RDM_ITEM_DATA.TARGET_ST_PERC] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Target OH EOL	Sum([RDM_ITEM_DATA.TARGET_INV_UNITS_OH_EOL])
Target ST Pct	Sum([RDM_ITEM_DATA.TARGET_ST_PERC] * [RDM_ITEM_DATA.COMMITTED_INV_UNITS]) / Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Temp MDs	Sum([RDM_ITEM_DATA.TEMP_MD_FLAG])
Total Cost Inv	Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS] * [RDM_ITEM_DATA.UNIT_COST])
Total Cost Inv NW	Sum([RDM_ITEM_DATA.FA_INVENTORY_UNITS_TW] * [RDM_ITEM_DATA.UNIT_COST])
Total EOW Inv	Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Total Inv	Sum([RDM_ITEM_DATA.COMMITTED_INV_UNITS])
Total Inv NW	Sum([RDM_ITEM_DATA.FA_INVENTORY_UNITS_NW])
Weekly Build LLLW	Sum([RDM_ITEM_DATA.SALES_UNITS_WM2]) / Sum([RDM_ITEM_DATA.SALES_UNITS_WM3])
Weekly Build LLW	Sum([RDM_ITEM_DATA.SALES_UNITS_WM1]) / Sum([RDM_ITEM_DATA.SALES_UNITS_WM2])
Weekly Build LW	Sum([RDM_ITEM_DATA.SALES_UNITS]) / Sum([RDM_ITEM_DATA.SALES_UNITS_WM1])
Weekly Build NW	Sum([RDM_ITEM_DATA.FA_SALES_UNITS_NW]) / Sum([RDM_ITEM_DATA.FA_SALES_UNITS_TW])
Weekly Build TW	Sum([RDM_ITEM_DATA.FA_SALES_UNITS_TW]) / Sum([RDM_ITEM_DATA.SALES_UNITS])
WOS	Sum([RDM_ITEM_DATA.WOS] * [RDM_ITEM_DATA.SALES_UNITS]) / Sum([RDM_ITEM_DATA.SALES_UNITS])

A

Standard Columns

This appendix contains a list of the standard columns that are included by default in the Markdown Optimization application. The information about each column listed in the following table includes the column key, default name, default description, calculation, DB column name, and function.

Table A–1 MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
HIERARCHY1	Hierarchy	1	HIERARCHY1	P4P_SAME_OR_NULL
HIERARCHY1_NAME	Hierarchy Name	1	HIERARCHY1_NAME	P4P_SAME_OR_NULL
HIERARCHY2	Hierarchy	1	HIERARCHY2	P4P_SAME_OR_NULL
HIERARCHY2_NAME	Hierarchy Name	1	HIERARCHY2_NAME	P4P_SAME_OR_NULL
HIERARCHY3	Hierarchy	1	HIERARCHY3	P4P_SAME_OR_NULL
HIERARCHY3_NAME	Hierarchy Name	1	HIERARCHY3_NAME	P4P_SAME_OR_NULL
HIERARCHY4	Hierarchy	1	HIERARCHY4	P4P_SAME_OR_NULL
HIERARCHY4_NAME	Hierarchy Name	1	HIERARCHY4_NAME	P4P_SAME_OR_NULL
HIERARCHY5	Hierarchy	1	HIERARCHY5	P4P_SAME_OR_NULL
HIERARCHY5_NAME	HierarchyName	1	HIERARCHY5_NAME	P4P_SAME_OR_NULL
HIERARCHY6	Hierarchy	1	HIERARCHY6	P4P_SAME_OR_NULL
HIERARCHY6_NAME	Hierarchy Name	1	HIERARCHY6_NAME	P4P_SAME_OR_NULL
HIERARCHY7	Hierarchy	1	HIERARCHY7	P4P_SAME_OR_NULL
HIERARCHY7_NAME	HierarchyName	1	HIERARCHY7_NAME	P4P_SAME_OR_NULL

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
HIERARCHY8	Hierarchy	1	HIERARCHY8	P4P_SAME_OR_NULL
HIERARCHY8_NAME	Hierarchy Name	1	HIERARCHY8_NAME	P4P_SAME_OR_NULL
HIERARCHY9	Hierarchy	1	HIERARCHY9	P4P_SAME_OR_NULL
HIERARCHY9_NAME	Hierarchy Name	1	HIERARCHY9_NAME	P4P_SAME_OR_NULL
HIERARCHY10	Hierarchy	1	HIERARCHY10	P4P_SAME_OR_NULL
HIERARCHY10_NAME	Hierarchy Name	1	HIERARCHY10_NAME	P4P_SAME_OR_NULL
HIERARCHY11	Hierarchy	1	HIERARCHY11	P4P_SAME_OR_NULL
HIERARCHY11_NAME	Hierarchy Name	1	HIERARCHY11_NAME	P4P_SAME_OR_NULL
HIERARCHY12	Hierarchy	1	HIERARCHY12	P4P_SAME_OR_NULL
HIERARCHY12_NAME	Hierarchy Name	1	HIERARCHY12_NAME	P4P_SAME_OR_NULL
HIERARCHY13	Hierarchy	1	HIERARCHY13	P4P_SAME_OR_NULL
HIERARCHY13_NAME	Hierarchy Name	1	HIERARCHY13_NAME	P4P_SAME_OR_NULL
HIERARCHY14	Hierarchy	1	HIERARCHY14	P4P_SAME_OR_NULL
HIERARCHY14_NAME	Hierarchy Name	1	HIERARCHY14_NAME	P4P_SAME_OR_NULL
HIERARCHY15	Hierarchy	1	HIERARCHY15	P4P_SAME_OR_NULL
HIERARCHY15_NAME	Hierarchy Name	1	HIERARCHY15_NAME	P4P_SAME_OR_NULL
HIERARCHY16	Hierarchy	1	HIERARCHY16	P4P_SAME_OR_NULL
HIERARCHY16_NAME	Hierarchy Name	1	HIERARCHY16_NAME	P4P_SAME_OR_NULL
HIERARCHY17	Hierarchy	1	HIERARCHY17	P4P_SAME_OR_NULL
HIERARCHY17_NAME	Hierarchy Name	1	HIERARCHY17_NAME	P4P_SAME_OR_NULL
firstReceiptDate/1st Rcpt Date	First receipt date.	0	FIRST_RECEIPT_DATE	P4P_MAX
lastReceiptDate/Last Rcpt Date	Last receipt date.	0	LAST_RECEIPT_DATE	P4P_MAX
vendorName/Vendor	Vendor name.	1	VENDOR_DESCRIPTION	P4P_SAME_OR_NULL

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
vendorID/Vendor #	Vendor number.	1	VENDOR	P4P_SAME_OR_NULL
startDate/startDate	The date of the start of the item's life in the application.	1	START_SELL_DATE	P4P_MIN
INT_OUT_OF_STOCK_DATE/Out Date	The target date by which the item should achieve its sell-thru.	0	OUT_OF_STOCK_DATE	P4P_MAX
INT_MOD_OUTDATE/New Out Date	New target date by which the item should achieve its sell-thru.	1	MODIFIED_OUT_OF_STOCK_DATE	P4P_MAX
seasonCode/Season Code	The item's season code.	1	SEASON_CODE	P4P_SAME_OR_NULL
INT_ORIG_RETAIL/Orig Rtl Price	The retail price at the start of life of the item which is typically the same as the original retail price.	1	ORIGINAL_RETAIL_PRICE	P4P_AVG
INT_UNIT_COST/AUC	The unit cost of the item.	1	UNIT_COST	P4P_AVG
markupPercInit/MU% Init	Initial markup percent is the percentage of the initial retail price that is markup above cost.	1	MARKUP_PERCENT	P4P_AVG
INT_ACCELERATED_SENDBACK_DATE/Sendback Date	Sendback date for accelerated markdown.	1	SENDBACK_DATE	P4P_MAX
INT_ACCELERATED_SENDBACK_DATE_SENT/Date Sent	The date the markdown was sent to the price change system.	1	SENT_DATE	P4P_MAX
INT_ACCELERATED_SENDBACK_PRICE/Price Sent	The price that was sent to the price change system.	1	SENT_MARKDOWN_PRICE	P4P_AVG
INT_ACCELERATED_SENDBACK_LADDER_ID/Ladder Sent	The price ladder that was sent to the price change system.	1	SENT_LADDER_ID	
INT_SALVAGE_VALUE/Salvage $	The $ salvage value for this item.	1	SALVAGE_VALUE	P4P_AVG
INT_SALVAGE_VAL_PERC/Salvage %	The salvage value for this item expressed as a % of original retail price.	1	SALVAGE_VALUE_PERC	P4P_MAX

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
projSalesUnitsTW/Proj Sls Units TW	The forecasted number of units that will be sold this week.	1	WEEKLY_ PROJECTED_UNIT_ SALES	P4P_SUM
projSalesDollarsTW/Proj Sls $ TW	The forecasted sales dollars this week.	1	WEEKLY_ PROJECTED_ DOLLAR_SALES	P4P_SUM
projSalesUnitsEOL/Proj Sls Units EOL	Projected total sales units for an item at the end of its life, assuming all application recommendations are taken.	1	XML_EOL_CUM_ UNIT_SALES	P4P_SUM
projSalesDollarsEOL/Proj Sales $ EOL	Projected total sales dollars for an item at the end of its life, assuming all application recommendations are taken.	1	XML_EOL_CUM_ DOLLARS_SALES	P4P_SUM
projSellThruEOL/Proj ST% EOL	Projected percent of total inventory quantity sold by the retailer specified out date, assuming all application recommendations are taken.	1	XML_ PROJSELLTHRUEOL	P4P_AVG
INT_TGT_SELLTHRU_ PERC/Target ST% EOL	Target sell-thru percent.	(1 - ENDING_ INVENTORY_PERC)	TGT_SELL_THRU_ PERC	P4P_MAX
INT_ENDING_INV/Target OH EOL	Target inventory remaining.	number	ENDING_ INVENTORY_UNITS	P4P_SUM
projOutDate/Proj Out Date	The projected date when the item will meet its sell-thru target.	date	PROJECTED_OUT_ OF_STOCK	P4P_MIN
projOnHandUnitsEOL/Proj OH EOL	Projected inventory units on hand at the retailer specified out date.	number	ENDING_ INVENTORY_UNITS	P4P_SUM
projRtlPriceEOL/Proj Rtl Price EOL	Project retail price at the out date.	value	REC_RTL_MIN	P4P_AVG
projOnHandRtlDollarsEOL /Proj OH Rtl $ EOL	Projected retail value of the inventory remaining at the out date.	projOnHandUnitsEOL * projRtlPriceEOL	PROJ_OH_RTL_EOL	P4P_SUM

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
projOnHandCostDollarsEOL/Proj OH Cost $ EOL	Projected cost of the inventory remaining at the out date.	projOnHandUnitsEOL * INT_UNIT_COST	PROJ_OH_COST_EOL	P4P_SUM
projMarginDollarsEOL/Proj GM$ EOL (Item)	The forecasted gross margin dollars over the life of an item (including salvage value).	projSalesDollarsEOL - (projSalesUnitsEOL + projOnHandUnitsEOL) * INT_UNIT_COST + salvValuePerc * INT_ORIG_RETAIL * projOnHandUnitsEOL	PROJ_STD_EOL_GM_AMOUNT	P4P_SUM
projMarginPercEOL/Proj GM% EOL (Item)	The forecasted gross margin percent over the life of an item (including salvage value).	projMarginDollarsEOL / projSalesDollarsEOL	PROJ_STD_EOL_GM_PERC	P4P_AVG
projMarginDollarsEOLGrp/Proj GM$ EOL (Grp)	The forecasted gross margin dollars over the life of an item in its pricing group context (including salvage value).	projSalesDollarsEOL - (projSalesUnitsEOL + projOnHandUnitsEOL) * INT_UNIT_COST + salvValuePerc * INT_ORIG_RETAIL * projOnHandUnitsEOL	PROJ_STD_EOL_GM_AMOUNT_C	P4P_SUM
projMarginPercEOLGrp/Proj GM% EOL (Grp)	The forecasted gross margin percent over the life of an item in its pricing group context (including salvage value).	projMarginDollarsEOL / projSalesDollarsEOL	PROJ_STD_EOL_GM_PERC_C	P4P_AVG
projSalesPriceTW/Proj Sls Price TW	The forecasted average selling price of units that will be sold this week.	value	WEEKLY_PROJECTED_SALES_PRICE	P4P_AVG
projSalesDollarsEOLGrp/Proj Sales $ EOL (Grp)	Projected total sales dollars for a group at the end of its life, assuming all application recommendations are taken.	EOL_CUM_DOLLARS_SALES_C + salesDollarsLTD	XML_EOL_CUM_DOLLARS_SALES_C	P4P_SUM
INT_TICKET_PRICE_WT_AVG/Rtl Price TW	The current retail price for the item this week (including any pending markdowns expected to be effective this week).	value	PERM_TICKET_PRICE	P4P_AVG

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
ownedRtlPriceTW/Owned Rtl	The current "owned at" price for the item this week (including any pending markdowns expected to be effective this week).	value	OWNED_RTL_PRICE	P4P_AVG
percOffOrigRtlTW/%Off TW	Current retail price as a percentage off of original retail.	1 - (INT_TICKET_PRICE / INT_ORIG_RETAIL)	CURRENT_ PERCENT_OFF	P4P_AVG
markupPercTW/MU% TW	Current markup percent is the percentage of this week's retail price that is markup above cost.	(INT_TICKET_PRICE - INT_UNIT_COST)/ INT_ TICKET_PRICE	XML_ MARKUPPERCTW	P4P_AVG
numStoresWithOnHandLW /# Stores OH	Number of stores with inventory on hand at the end of last week.	number	NO_STORE_WITH_ ON_HAND	P4P_SUM
onHandUnitsLW/On Hand	Inventory on hand in the stores as of the beginning of this week (end of last week).	number	CURRENT_UNITS_ ON_HAND	P4P_SUM
onHandUnitsLLW/On Hand LLW	Inventory on hand in the stores as of the end last week minus one week.	number	WEEK_MINUS_1_ UNITS_ON_HAND	P4P_SUM
onHandUnitsLLLW/On Hand LLLW	Inventory on hand in the stores as of the end last week minus two weeks.	number	WEEK_MINUS_2_ UNITS_ON_HAND	P4P_SUM
onHandUnitsLLLLW/On Hand LLLLW	Inventory on hand in the stores as of the end last week minus three weeks.	number	WEEK_MINUS_3_ UNITS_ON_HAND	P4P_SUM
onOrderUnitsLW/On Order	Inventory that is in transit to or otherwise committed to the stores as of the beginning of this week (end of last week).	number	CURRENT_UNITS_ ON_ORDER	P4P_SUM

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
dcOnHandUnitsLW/DC On Hand	Inventory on hand in the DC location that is not yet committed to any store as of the beginning of this week (end of last week).	number	WAREHOUSE_ON_ ORDER	P4P_SUM
dcOnOrderUnitsLW/DC On Order	Inventory that is on order to the DC location as of the beginning of this week (end of last week).	number	WAREHOUSE_ON_ HAND	P4P_SUM
onHandRtlDollarsLW/OH Rtl $	Retail value of the inventory on hand in the stores.	onHandUnitsLW * rtlPriceLW	XML_CURRENT_ ON_HAND_ DOLLAR	P4P_SUM
onOrderRtlDollarsLW/OO Rtl $	Retail value of the inventory that is in transit or otherwise committed to the stores.	onOrderUnitsLW * rtlPriceLW	XML_CURRENT_ ON_ORDER_ DOLLAR	P4P_SUM
onHandCostDollarsLW/OH Cost $	Cost of the inventory on hand in the stores.	onHandUnitsLW * INT_ UNIT_COST	XML_ ONHANDCOSTDOL LARS	P4P_SUM
onOrderCostDollarsLW/O O Cost $	Cost of the inventory that is in transit or otherwise committed to the stores.	onOrderUnitsLW * INT_ UNIT_COST	XML_ ONORDERCOSTDOL LARS	P4P_SUM
INT_INVENTORY/Total Inv	The remaining inventory commitment for the item including store and DC on hand and on order units.	dcOnHandUnitsLW + dcOnOrderUnitsLW + onHandUnitsLW + onOrderUnitsLW	COMMITTED_INV_ UNITS	P4P_SUM
weeksOfSupplyLW/WOS	The weeks of supply on hand at the start of this week.	onHandUnitsLW / salesUnitsLW	XML_WEEKS_OF_ SUPPLY	P4P_AVG
promoFlag/Promo Flag	This is a flag to identify items that have a planned promotion this week or in the future.	number	PROMOTION_FLAG	P4P_MAX
promoRtlLowest/Promo Rtl Lowest	Lowest planned promo price for the item at any point in the future.	value	LOWEST_FUTURE_ PROMOTES_PRICE	P4P_MIN

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
projOnHandUnitsEffDt/Proj OH Eff Dt	The projected inventory level on hand in the stores at the recommended effective date.	Does not include DC OH or DC OO	PROJ_OH_UNITS_EFF_DT	P4P_SUM
INT_PROPOSED_PRICE/MD Price	Drop-down for selecting markdown price.	The dropdown will default to the Rec MD Price	PROPOSED_PRICE	
INT_LADDER_ID/MD Type	Drop-down for selecting price ladder.	text	LADDER_ID	
markdownNumber/MD #	The number of past MD's +1.	number	MARKDOWN_NUMBER	P4P_SAME_OR_NULL
OPPTY_COST/Oppy Cost	This is the opportunity cost (margin loss) of deferring taking a recommended item-context markdown until the next available markdown date assuming that the deferred markdown will then be taken as recommended (for item-context) by the application.	value	OPPTY_COST	P4P_SUM
INT_RECOMMENDED_RETAIL/Rec Rtl	Recommended markdown price.	value	RECOMMENDED_RETAIL_PRICE	P4P_AVG
recPercOffOrigRtl/Rec %Off Orig	Recommended markdown price as a percentage off of original retail.	(INT_ORIG_RETAIL - INT_RECOMMENDED_RETAIL) / INT_ORIG_RETAIL	XML_RECPERCOFFORIGRTL	P4P_AVG
recPercOffCurrRtl/Rec %Off Curr	Recommended markdown price as a percentage off of current retail.	(INT_TICKET_PRICE - INT_RECOMMENDED_RETAIL) / INT_TICKET_PRICE	XML_RECPERCOFFCURRRTL	P4P_AVG
takenPercOffOrigRtl/Taken %Off Orig	Taken markdown price as a percentage off of original retail.	If markdown_flag = 'y' Then (INT_ORIG_RETAIL - takenRtlPrice) / INT_ORIG_RETAIL	XML_TAKENPERCOFFORIGRTL	P4P_AVG
takenPercOffCurrRtl/Taken %Off Curr	Taken markdown price as a percentage off of current retail.	If markdown_flag = 'y' Then (INT_TICKET_PRICE - takenRtlPrice) / INT_TICKET_PRICE	XML_TAKENPERCOFFCURRRTL	P4P_AVG
INT_TAKEN_PRICE/Taken Rtl	Taken markdown price.	If markdown_flag = 'y' Then TAKEN_PRICE	TAKEN_PRICE	P4P_AVG

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
effectiveDate/Effective Date	Markdown effective date.	date	EFFECTIVE_DATE	P4P_MIN
nextRecMDDate/Next Rec Date	Next recommended markdown date after the current markdown effective date.	date	PROJECTED_NEXT_MARKDOWN	P4P_MIN
nextRecRtlPrice/Next Rec Rtl	Next recommended markdown price after the current markdown effective date.	value	NEXT_RTL_PRICE	P4P_MIN
nextRecPercOffOrigRtl/Next Rec %Off	Next recommended markdown price after the current markdown effective date as a percentage off of original retail.	(INT_ORIG_RETAIL - NextRecRtlPrice) / INT_ORIG_RETAIL	NEXT_REC_PERC_OFF_ORRTL	P4P_MAX
INT_REC_MD_COST/Rec MD $	Markdown cost (using retail accounting) of taking the recommended markdown.	projOnHandUnitsTW * (ownedRtlPriceTW - INT_RECOMMENDED_RETAIL)	INT_REC_MD_COST	P4P_SUM
recMDDollarsCost/Rec MD $ Cost	The cost of the inventory recommended for markdown.	For recommended items: projOnHandUnitsTW * INT_UNIT_COST	XML_RECMDDOLLARSCOST	P4P_SUM
INT_TAKEN_MD_COST/Taken MD $	Markdown cost (using retail accounting) of the taken markdown.	projOnHandUnitsTW * (ownedRtlPriceTW - takenRtlPrice)	INT_TAKEN_MD_COST	P4P_SUM
takenMDDollarsCost/Taken MD $ Cost	The cost of the inventory taken for markdown.	For taken items: projOnHandUnitsTW * INT_UNIT_COST	XML_TAKENMDDOLLARSCOST	P4P_SUM
recRetailPriceMin/Rec Rtl Min	The minimum recommended retail price for the items underneath.	MIN(INT_RECOMMENDED_RETAIL)	RECOMMENDED_RETAIL_PRICE	P4P_MIN
recRetailPriceMax/Rec Rtl Max	The maximum recommended retail price for the items underneath.	MAX(INT_RECOMMENDED_RETAIL)	RECOMMENDED_RETAIL_PRICE	P4P_MAX

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
INT_RECOMMENDED_ ITEM_FLAG/# Item Recs	Shows 1 for a recommended item, 0 for a non-recommende d item. Summary rows show the total number of recommended items underneath.	number	RECOMMENDED_ ITEM_FLAG	P4P_SUM
numItems/# Items	For summary rows only- shows the number of items underneath.	1number	XML_NUMITEMS	P4P_SUM
MD_FLAG_VIEW/Taken MD	Markdown status of the item.	If MARKDOWN_FLAG = 1 Then 'Taken Item Rec' Else If MARKDOWN_ FLAG = 2 Then 'Taken Pricing Grp Rec' Else If MARKDOWN_FLAG = 3 Then 'Taken Modified' Else If MARKDOWN_ FLAG = 4 Then 'Taken Budget Const.' Else 'Not Taken'	MARKDOWN_FLAG	P4P_MAP_ STATUS_ AND_ SAME_OR_ TEMPLATE
numTakenMarkdowns/# Taken MD	Shows 1 for an item with a taken markdown, 0 for an item without a taken markdown. Summary rows show the total number of items with taken markdowns.	If MARKDOWN_FLAG = 'y' Then 1 Else 0	XML_ NUMRECTAKEN	P4P_SUM
GRP_OPPTY_COST/Grp Oppy Cost	This is the opportunity cost (margin loss) of deferring taking a recommended group-context markdown until the next available markdown date assuming that the deferred markdown will then be taken as recommended (for group context) by the application.	value	GRP_OPPTY_COST	P4P_SUM
INT_COLLECTION_ RECOMMENDED_ PRICE/Grp Rec Rtl	Recommended markdown price if item is optimized as part of a pricing group.	value	COLLECTION_ RECOMMENDED_ PRICE	P4P_AVG

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
INT_REC_COLLECTION_FLAG/# Grp Recs	Shows 1 for an item with a group recommendation, 0 for items without. Summary rows show the total number of recommended items underneath.	number	RECOMMENDED_COLLECTION_FLAG	P4P_SUM
RevenueLostBudgetConst/GM $ (Optimal) - GM$ (Budget Constrained)	Amount of margin forced to be forfeited by limited markdown budget.	value	REVENUE_LOST_BUDGET_CONST	P4P_SUM
salesDollarsLLLLW/Sls $ LLLLW	The total dollar sales four weeks ago (last week minus three weeks).	value	DOLLAR_SALES_WEEK_MINUS_3	P4P_SUM
sellThruLLLLW/ST% LLLLW	The number of units sold 4 weeks ago divided by the units on hand at the start of the week.	salesUnitsLLLLW / (onHandUnitsLLLLW + salesUnitsLLLLW)	SELL_THRUPERCENT_WEEK_MINUS_3	P4P_AVG
salesUnitsLLLLW/Sls Units LLLLW	The number of units sold four weeks ago (last week minus three weeks).	number	UNIT_SALES_WEEK_MINUS_3	P4P_SUM
salesDollarsLLLW/Sls $ LLLW	The total dollar sales three weeks ago (last week minus two weeks).	value	DOLLAR_SALES_WEEK_MINUS_2	P4P_SUM
sellThruLLLW/ST% LLLW	The number of units sold 3 weeks ago divided by the units on hand at the start of the week.	salesUnitsLLLW / (onHandUnitsLLLW + salesUnitsLLLW)	SELL_THRUPERCENT_WEEK_MINUS_2	P4P_AVG
salesUnitsLLLW/Sls Units LLLW	The number of units sold three weeks ago (last week minus two weeks).	number	UNIT_SALES_WEEK_MINUS_2	P4P_SUM
salesDollarsLLW/Sls $ LLW	The total dollar sales two weeks ago (last week minus one week)	value	DOLLAR_SALES_WEEK_MINUS_1	P4P_SUM

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
sellThruLLW/ST% LLW	The number of units sold 2 weeks ago divided by the units on hand at the start of the week.	salesUnitsLLW/(onHand UnitsLLW + salesUnitsLLW)	SELL_ THRUPERCENT_ WEEK_MINUS_1	P4P_AVG
salesUnitsLLW/Sls Units LLW	The number of units sold two weeks ago (last week minus one week).	number	UNIT_SALES_ WEEK_MINUS_1	P4P_SUM
salesDollarsLW/Sls $ LW	The total dollar sales last week.	value	DOLLAR_SALES_ THROUGH_WEEK	P4P_SUM
sellThruLW/ST% LW	The number of units sold last week divided by the units on hand at the start of the week.	salesUnitsLW/(onHandU nitsLW + salesUnitsLW)	SELL_ THRUPERCENT_ CURRENT_WEEK	P4P_AVG
salesUnitsLW/Sls Units LW	The number of units sold last week.	number	UNIT_SALES_ THROUGH_WEEK	P4P_SUM
INT_MOST_RECENT_ RETAIL/Rtl Price LW	The retail price for the item as of the end of last week (start of this week).	value	CURRENT_RETAIL_ PRICE	P4P_AVG
averRtlPriceLW/AUR LW	The average unit retail price of the item last week.	salesDollarsLW / salesUnitsLW	AVERAGE_PRICE	P4P_AVG
INT_STD_AVG_ PRICE/AUR LTD	The average unit retail price of the item life to date.	salesDollarsLTD / salesUnitsLTD	STD_AVERAGE_ PRICE	P4P_AVG
INT_CUM_SALES/Sls Units LTD	The total number of units sold since the item's Start Sell Date.	Sum of LW Unit Sls from each weekly retailer data feed for all weeks from the Start Date	CUMULATIVE_ QUANTITY_SOLD	P4P_SUM
salesDollarsLTD/Sls $ LTD	The total sales dollars since the item's Start Sell Date.	Sum of LW Sls $ from each weekly retailer data feed for all weeks from the Start Date or First Fiscal Week of the Season, whichever is later	CUMULATIVE_ SALES_DOLLARS	P4P_SUM
INT_STD_SELLTHRU_ PERC/ST% LTD	This is the percent of store inventory sold from the beginning of the life up to the current date.	100 * ((TD Sls Units) / (TD Sls Units + EOP Inv TTT + SOO)	CUMULATIVE_ SELLTHRU_ PERCENT	P4P_AVG
marginDollarsLTD/GM$ LTD	Life to Date gross margin dollars.	salesDollarsLTD - (salesUnitsLTD *INT_ UNIT_COST)	CUMM_GROSS_ PROFIT_DOLLAR	P4P_SUM

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
marginPercLTD/GM% LTD	Life to Date gross margin %.	marginDollarsLTD / salesDollarsLTD	CUMM_GROSS_ PROFIT_PERC	P4P_AVG
totalInventoryInitDollarsLW /Total Inv Init$ LW	REQUIRED FOR AGGREGATION	value	XML_ TOTALINVINITDOL LARSLW	P4P_SUM
totalInventoryRtlDollarsTW /Total Inv Rtl$ TW	REQUIRED FOR AGGREGATION	value	XML_ TOTALINVRTLDOLL ARSTW	P4P_SUM
totalUnitsLW/Total Units LW	REQUIRED FOR AGGREGATION	number	XML_ TOTALUNITSLW	P4P_SUM
totalUnitsLLW/Total Units LLW	REQUIRED FOR AGGREGATION	number	XML_ TOTALUNITSLLW	P4P_SUM
totalUnitsLLLW/Total Units LLLW	REQUIRED FOR AGGREGATION	number	XML_ TOTALUNITSLLLW	P4P_SUM
totalUnitsLLLLW/Total Units LLLLW	REQUIRED FOR AGGREGATION	number	XML_ TOTALUNITSLLLLW	P4P_SUM
totalUnitsLTD/Total Units LTD	REQUIRED FOR AGGREGATION	number	XML_ TOTALUNITSLTD	P4P_SUM
INT_ITEM_ REGION/locHierarchy1	locHierarchy1	text	REGION	P4P_SAME_ OR_NULL
INT_ITEM_REGION_ DESCRIPTION/locHierarch y1Name	locHierarchy1Na me	text	REGION_ DESCRIPTION	P4P_SAME_ OR_NULL
INT_COLLECTION_REC_ MD_COST/Grp Rec MD $	Markdown cost (using retail accounting) of taking the recommended group markdown.	value	INT_COLLECTION_ REC_MD_COST	P4P_SUM
INT_COLLECTION_ NAME/Grp Desc.	Group Description	text		P4P_SAME_ OR_ TEMPLATE
INT_MOD_INV_TARGET_ ST_PERC/New Sell Thru %	New Sell Thru %	number		P4P_MAX
INT_MOD_INV_TARGET_ END_UNITS/New End Inv	New End Inv	number		P4P_SUM
INT_CURR_PERC_OFF_ ORIG/Curr % Off	Curr % Off	value		P4P_MAX
INT_STYLE_DESC/ HIERARCHY6_NAME	HIERARCHY6_ NAME	This is set to the hierarchy level that will display the hyperlink for the item pop-up.		P4P_SAME_ OR_NULL
projSalesUnitsEOLandEIU/ Total Buy	The number of units sold life to date plus the forecastable (committed) inventory.	projSalesUnitsEOL + + projOnHandUnitsEOL	XML_PROJ_SLS_ UNITS_EOL_EIU	P4P_SUM

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
projSalesUnitsEOLGrp/Proj Sls Units EOL (Grp)	Projected total sales units for a group item at the end of its life, assuming all application recommendations are taken.	number	XML_EOL_CUM_UNIT_SALES_GRP	P4P_SUM
projSellThruEOLGrp/Proj ST% EOL (Grp)	Projected percent of total inventory quantity sold by the retailer specified out date, assuming all application group recommendations are taken.	number	XML_PROJSELLTHRUEOLGRP	P4P_AVG
projOnHandUnitsEOLgrp/Proj OH EOL (Grp)	Projected inventory units on hand at the retailer specified out date for an item as part of a price group.	number	ENDING_INVENTORY_UNITS_C	P4P_SUM
projOnHandCostDollarsEOLgrp/ Proj OH Cost $ EOL (Grp)	Projected cost of the inventory remaining at the out date for an item as part of a price group.	value	PROJ_OH_COST_EOL_GRP	P4P_SUM
projSalesUnitsEOLandEIUgrp/Total Buy (Grp)	The number of units sold life to date plus the forecastable (committed) inventory for an item as part of a price group.	projSalesUnitsEOLGrp + projOnHandUnitsEOLgrp	XML_PROJ_SLS_UNITS_EOL_EIU_GRP	P4P_SUM
grpRecPercOffOrigRtl/Grp Rec %Off Orig	Group recommended markdown price as a percentage off of original retail.	(INT_ORIG_RETAIL - INT_COLLECTION_RECOMMENDED_PRICE) / INT_ORIG_RETAIL	XML_GRPRECPERCOFFORIGRTL	P4P_AVG
grpRecPercOffCurrRtl/Grp Rec %Off Curr	Group recommended markdown price as a percentage off of current retail.	(INT_TICKET_PRICE -INT_COLLECTION_RECOMMENDED_PRICE) / INT_TICKET_PRICE	XML_GRPRECPERCOFFCURRRTL	P4P_AVG
IATTRIBUTE1/IATTRIBUTE1	IATTRIBUTE1	text	IATTRIBUTE1	P4P_SAME_OR_NULL
IATTRIBUTE1_DATE/IATTRIBUTE1_DATE	IATTRIBUTE1_DATE	date	IATTRIBUTE1_DATE	IATTRIBUTE1_DATE

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
IATTRIBUTE1_ NUMBER/IATTRIBUTE1_ NUMBER	IATTRIBUTE1_ NUMBER	number	IATTRIBUTE1_ NUMBER	P4P_SUM
IATTRIBUTE2/IATTRIBUTE2	IATTRIBUTE2	text	IATTRIBUTE2	P4P_SAME_ OR_NULL
IATTRIBUTE2_ DATE/IATTRIBUTE2_ DATE	IATTRIBUTE2_ DATE	date	IATTRIBUTE2_DATE	P4P_MIN
IATTRIBUTE2_ NUMBER/IATTRIBUTE2_ NUMBER	IATTRIBUTE2_ NUMBER	number	IATTRIBUTE2_ NUMBER	P4P_SUM
IATTRIBUTE3/IATTRIBUTE3	IATTRIBUTE3	text	IATTRIBUTE3	P4P_SAME_ OR_NULL
IATTRIBUTE3_ DATE/IATTRIBUTE3_ DATE	IATTRIBUTE3_ DATE	date	IATTRIBUTE3_DATE	IATTRIBUTE1_DATE
IATTRIBUTE3_ NUMBER/IATTRIBUTE3_ NUMBER	IATTRIBUTE3_ NUMBER	number	IATTRIBUTE3_ NUMBER	P4P_SUM
IATTRIBUTE4/IATTRIBUTE4	IATTRIBUTE4	text	IATTRIBUTE4	P4P_SAME_ OR_NULL
IATTRIBUTE4_ DATE/IATTRIBUTE4_ DATE	IATTRIBUTE4_ DATE	date	IATTRIBUTE4_DATE	IATTRIBUTE1_DATE
IATTRIBUTE4_ NUMBER/IATTRIBUTE4_ NUMBER	IATTRIBUTE4_ NUMBER	number	IATTRIBUTE4_ NUMBER	P4P_SUM
IATTRIBUTE5/IATTRIBUTE5	IATTRIBUTE5	text	IATTRIBUTE5	P4P_SAME_ OR_NULL
IATTRIBUTE5_ DATE/IATTRIBUTE5_ DATE	IATTRIBUTE5_ DATE	date	IATTRIBUTE5_DATE	IATTRIBUTE1_DATE
IATTRIBUTE5_ NUMBER/IATTRIBUTE5_ NUMBER	IATTRIBUTE5_ NUMBER	number	IATTRIBUTE5_ NUMBER	P4P_SUM
IATTRIBUTE6/IATTRIBUTE6	IATTRIBUTE6	text	IATTRIBUTE6	P4P_SAME_ OR_NULL
IATTRIBUTE6_ DATE/IATTRIBUTE6_ DATE	IATTRIBUTE6_ DATE	date	IATTRIBUTE6_DATE	IATTRIBUTE1_DATE
IATTRIBUTE6_ NUMBER/IATTRIBUTE6_ NUMBER	IATTRIBUTE6_ NUMBER	number	IATTRIBUTE6_ NUMBER	P4P_SUM
IATTRIBUTE7/IATTRIBUTE7	IATTRIBUTE7	text	IATTRIBUTE7	P4P_SAME_ OR_NULL
IATTRIBUTE7_ DATE/IATTRIBUTE7_ DATE	IATTRIBUTE7_ DATE	date	IATTRIBUTE7_DATE	IATTRIBUTE1_DATE

Table A–1 (Cont.) MDO Standard Columns

Column Key/Default Name	Default Description	Calculation	DB Column Name	Function
IATTRIBUTE7_NUMBER/IATTRIBUTE7_NUMBER	IATTRIBUTE7_NUMBER	number	IATTRIBUTE7_NUMBER	P4P_SUM
IATTRIBUTE8/IATTRIBUTE8	IATTRIBUTE8	text	IATTRIBUTE8	P4P_SAME_OR_NULL
IATTRIBUTE8_DATE/IATTRIBUTE8_DATE	IATTRIBUTE8_DATE	date	IATTRIBUTE8_DATE	IATTRIBUTE1_DATE
IATTRIBUTE8_NUMBER/IATTRIBUTE8_NUMBER	IATTRIBUTE8_NUMBER	number	IATTRIBUTE8_NUMBER	P4P_SUM